TPC Benchmark™ H Full Disclosure Report
for
SGI Altix 350
using
IBM DB2 Universal Database V8.2

Submitted for Review

2005/03/25

First Edition – 2005/03/25

THE INFORMATION CONTAINED IN THIS DOCUMENT IS DISTRIBUTED ON AN AS IS BASIS WITHOUT ANY WARRANTY EITHER EXPRESSED OR IMPLIED. All performance information contained in this report was obtained in a controlled environment. Results obtained in other operating environments may vary significantly.

This publication was produced in the United States. SGI 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 SGI representative for information on products and services available in your area.

© Copyright SGI 2005. 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.

Trademarks

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

Trademark of Silicon Graphics, Inc.:
SGI, the SGI logo, Altix, SGI Altix 350 SGI FullExpress, SGI InfiniteStorage, SGI TP9300, SGI TP9500, Silicon Graphics

Trademark of International Business Machines Corporation:
IBM, the IBM logo, DB2, DB2 Universal Database

Trademark of Intel Corporation:
Intel, Itanium 2, Itanium

Trademark of the Transaction Processing Performance Council:
TPC Benchmark, TPC-H, QppH, QthH, QphH

Trademark of Linus Torvalds:
Linux

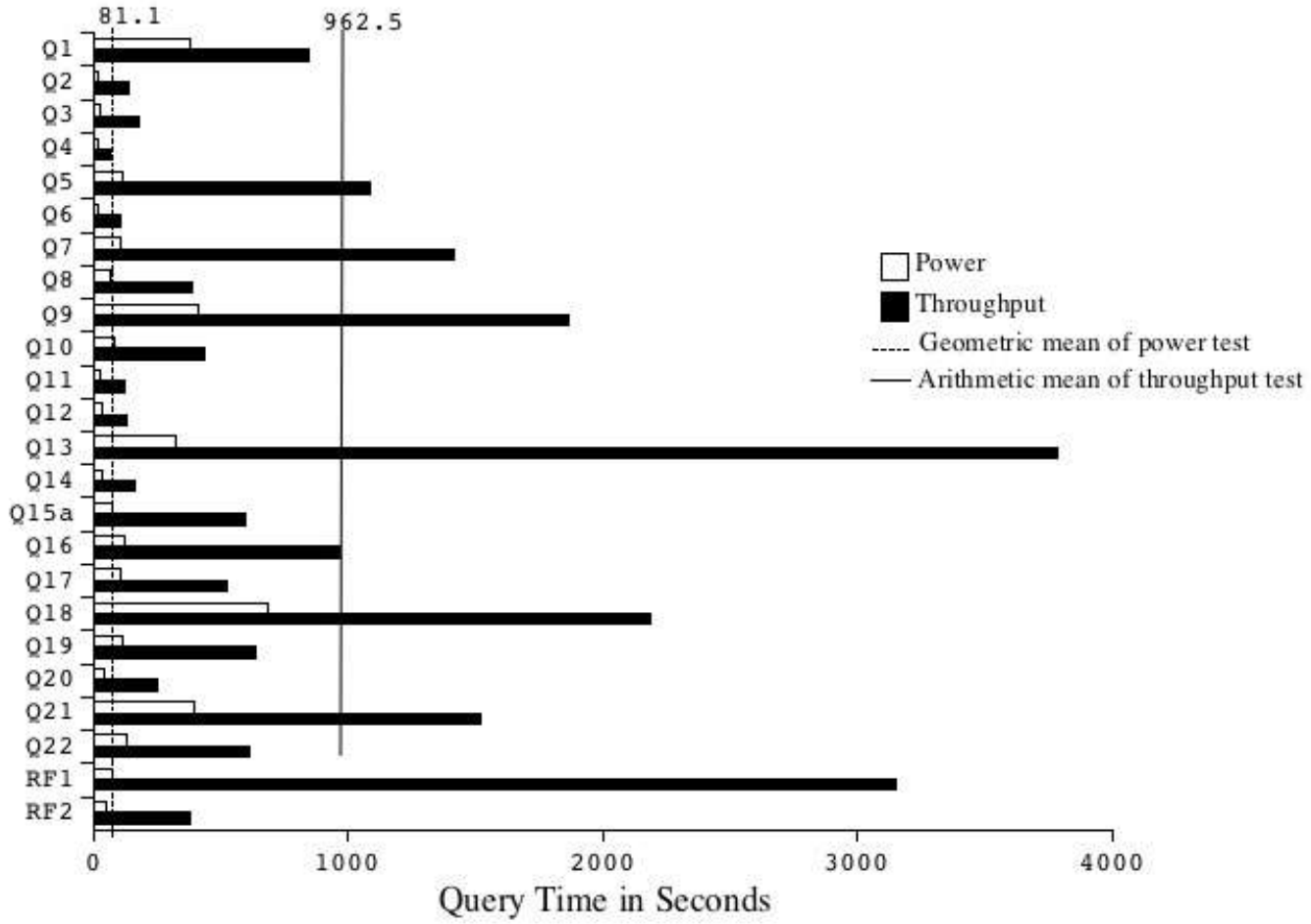
Trademark of SUSE AG, a Novell business:
SUSE, the SUSE logo

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

sgi	**SGI Altix 350** **with** **DB2 UDB V8.2**	**TPC-H Revision 2.1.0** **Report Date: 2005/03/25**

Total System Cost	Composite Query-per-hour Metric	Price/Performance
$586,017	**9962.5 QphH@300GB**	**$59/QphH@300GB**

Database Size	Database Manager	Operating System	Other Software	Availability Date
300GB	DB2 UDB V8.2 + FixPak 1	SUSE Linux Enterprise Server 9 + SP1	None	2005/03/25



Database Load Time: 2:45:14	Load Included Backup: Y	Total Data Storage / Database Size: 17.03
RAID (Base Table): N	RAID (Base Table and Auxiliary Data Structures): N	RAID (ALL): N

System Configuration: SGI Altix 350

Processors: 8 x 1.6GHz, 6MB-L3 Intel Itanium 2 Processors

Memory: 64GB

Disk Controllers: 8 x Dual-port 2Gbit Fibre Channel HBAs

Disks: 70 x 73GB FC disks

SGI Altix 350
with
DB2 UDB V8.2

TPC-H Revision 2.1.0

Report Date: 2005/03/25

Description	Part #	Src	Unit Price	Qty	Extended Price	3-Yr. Maint. Price
Server Hardware						
SGI Altix 350 System	LS-SYS-A350	1	$0	1	$0	$7,620
SGI Altix 350 short rack	S-RACK-A350	1	$3,500	1	$3,500	
SGI Altix 350 1.6GHz, 6MB-L3 processor	LS-CPU-A350-96	1	$6,000	8	$48,000	$6,968
SGI Altix 350 8GB DIMM set (4x2GB)	LS-MEM-A350-8G	1	$12,900	8	$103,200	
73GB, 10K SCSI disk	LS-SCSIDRV-73GB	1	$1,400	2	$2,800	
SGI Altix 350 Base System	LS-BASE-A350	1	$5,000	1	$5,000	
IO9 PCI card s/SCSI	LS-BASEIO-SCSI	1	$1,000	1	$1,000	
SGI Altix 350 CPU, memory & PCI expansion module	LS-EXP-CMPX-C	1	$4,500	3	$13,500	
PDU220V	RAKPWR220V	1	$0	1	$0	
Rackmount shelves. Factory installed.	RM-KIT-SHELVES-I	1	$300	4	$1,200	
500W power supply	RPS-A350	1	$500	4	$2,000	
10 metre 50 micron optical cable w/LC connectors	X-F22-OPT-10M	1	$223	16	$3,568	
SUSE Linux Ent. Server 9 Maint. 3Yr/1-8cpu	SR5-SLES9-SVR16P-3Y	1	$3,064	1	$3,064	
SUSE Linux Ent. Server 9 for IPF Media kit	SC5-SLES9-ENG-GER	1	$33	1	$33	
Dual channel optical Fibre Channel HBA	PCIX-FC-2POPT-B	1	$4,600	8	$36,800	
CD-ROM Media Update requirement	MO5-CD	1	$0	1	$0	$972
				Server Hardware Subtotals:	$223,665	$15,560
Server Storage						
SGI TP9300	TP9300-BASE-2C	1	$18,750	8	$150,000	$50,576
2Gbit, 73GB, 15K rpm LP disk drive	TP9500-73GB-E	1	$1,175	68	$79,900	
38-U rack enclosure fr TP9500/TP9300	TP9500-RACK	1	$8,990	1	$8,990	
U.S. Power desination kit	TP9500-DK-US	1	$250	1	$250	
TP9300 destination kit for USA/Canada/Japan	DK-TP-001	1	$0	1	$0	
TPSSM host S/W license req'd for TP9300	SC5-TPSSM-9.10	1	$3,750	8	$30,000	
				Server Storage Subtotals	$269,140	$50,576
				Server Hardware + Server Storage Subtotals:	$498,052	$66,136
Software Licenses						
DB2 UDB Enterprise Server Edition (ESE) V8.2 License/1-yr. Maint.		2	$22,260	8	$178,080	
DB2 UDB Enterprise Server Edition (ESE) V8.2 Support - 1-yr./proc.		2	$1,060	16		$16,960
DB2 UDB Data Partitioning Feature (DPF) Lic./1-yr. Maint.		2	$6,686	8	$53,488	
DB2 UDB Data Partitioning Feature (DPF) Support 1 yr/proc		2	$318	16		$5,088
				Software Licenses Subtotals:	$231,568	$22,048
				Totals:	$724,373	$88,184
				SGI Large System Discount:		-$226,540
				Total Three-Year Cost of Ownership:		$586,017
				QphH@300GB		9,962.5
				US$/QphH@300GB		$59

Sources: 1 = SGI – Scott Ellman <scotte@sgi.com> 1-650-933-4577; 2 = IBM– Bernard Spang <spang@us.ibm.com> 1-914-766-1491

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 section of the TPC benchmark specifications.

If you find that the stated prices are not available according to these terms, please inform the TPC at <pricing@tpc.org>. Thank you.

Measurement Results:

Database Scale Factor	300
Total Data Storage/Database Size	17.03
Start of Database Load	03:45:35
End of Database Load	06:30:49
Database Load Time	02:45:14
Query Streams for Throughput Test	6
TPC-H Power	13,311.4
TPC-H Throughput	7,456.1
TPC-H Composite Query-per-Hour (QphH@300GB)	9,962.5
Total System Price over 3 Years	$586,017
TPC-H Price/Performance Metric ($/QphH@300GB)	$59

Measurement Interval:

Measurement Interval in Throughput Test (Ts) = 19120 seconds

Duration of Stream Execution:

	Seed	Query Start Date/Time	RF1 Start Date/Time	RF2 Start Date/Time	Duration
		Query End Date/Time	RF1 End Date/Time	RF2 End Date/Time	
Stream-00		02/22/05 11:40:23	02/22/05 11:39:12	02/22/05 12:35:19	
	222063049	02/22/05 12:35:19	02/22/05 11:40:23	02/22/05 12:36:07	00:56:55
Stream-01		02/22/05 12:39:22	02/22/05 12:39:27	02/22/05 17:49:45	
	22063050	02/22/05 17:44:00	02/22/05 17:49:45	02/22/05 17:50:22	05:04:38
Stream-02		02/22/05 12:39:22	02/22/05 17:50:22	02/22/05 17:51:25	
	22063051	02/22/05 17:34:27	02/22/05 17:51:25	02/22/05 17:52:04	04:55:05
Stream-03		02/22/05 12:39:22	02/22/05 17:52:04	02/22/05 17:52:56	
	22063052	02/22/05 17:31:13	02/22/05 17:52:56	02/22/05 17:53:35	04:51:51
Stream-04		02/22/05 12:39:22	02/22/05 17:53:35	02/22/05 17:54:28	
	22063053	02/22/05 17:35:33	02/22/05 17:54:28	02/22/05 17:55:03	04:56:11
Stream-05		02/22/05 12:39:22	02/22/05 17:55:03	02/22/05 17:55:57	
	22063054	02/22/05 17:41:27	02/22/05 17:55:57	02/22/05 17:56:33	05:02:05
Stream-06		02/22/05 12:39:22	02/22/05 17:56:33	02/22/05 17:57:24	
	22063055	02/22/05 17:48:33	02/22/05 17:57:24	02/22/05 17:58:02	05:09:01

TPC-H Timing Intervals (in seconds):

Query	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
Stream 00	374.5	19.6	26.8	19.2	109.7	16.3	103.6	66.6	410.0	81.1	28.2	34.1
Stream 01	771.0	212.5	146.3	37.5	1133.0	115.9	1975.4	502.6	1952.1	573.0	118.7	154.6
Stream 02	635.0	138.4	105.1	47.5	920.1	141.6	1199.1	318.5	2141.4	455.8	128.1	167.9
Stream 03	677.8	108.8	242.2	99.2	1300.7	65.2	1222.9	423.3	1580.7	320.3	78.4	142.3
Stream 04	813.4	161.1	233.1	62.2	1129.2	49.3	1389.0	326.4	2223.0	416.6	140.6	85.7
Stream 05	1067.4	115.6	212.6	81.8	977.4	95.1	1194.5	292.5	1695.7	507.3	107.6	109.9
Stream 06	1088.2	84.7	121.4	45.2	1068.6	139.8	1521.6	472.6	1595.1	353.0	151.4	119.9
Minimum	635.0	84.7	105.1	37.5	920.1	49.3	1194.4	292.5	1580.7	320.3	78.4	85.7
Average	842.1	136.9	176.8	62.2	1088.2	101.1	1417.0	389.3	1864.7	437.7	120.8	130.1
Maximum	1088.2	212.5	242.2	99.2	1300.7	141.6	1975.3	502.5	2222.9	573.0	151.4	167.9

Stream ID	Q13	Q14	Q15a	Q16	Q17	Q18	Q19	Q20	Q21	Q22	RF1	RF2
Stream 00	321.6	30.0	71.5	120.3	101.3	681.8	115.8	38.2	397.0	128.7	71.0	48.3
Stream 01	3151.4	67.9	545.8	953.8	521.9	1836.3	656.7	235.4	1714.9	901.9	18618.3	37.5
Stream 02	3477.5	221.2	945.0	1261.4	625.0	1758.6	704.7	157.6	1696.4	459.9	62.5	38.8
Stream 03	3600.7	195.8	725.3	962.4	512.6	2346.2	432.7	130.2	1756.8	587.0	52.8	38.2
Stream 04	3444.1	57.4	473.4	1061.0	459.3	2433.1	633.0	297.3	1454.7	428.2	53.1	35.7
Stream 05	4270.0	201.9	467.0	811.3	433.3	2479.8	613.8	148.0	1719.4	523.4	53.4	36.1
Stream 06	4762.8	226.6	422.9	716.3	605.6	2251.3	759.1	525.7	768.6	751.6	50.6	38.4
Minimum	3151.4	57.4	422.9	716.3	433.3	1758.6	432.7	130.2	768.6	428.2	50.6	35.7
Average	3784.4	161.8	596.6	961.0	526.3	2184.2	633.3	249.0	1518.5	608.7	3148.4	37.5
Maximum	4762.8	226.6	945.0	1261.4	625.0	2479.8	759.1	525.6	1756.8	901.9	18618.3	38.8

Table of Contents

Preface....10
1General Items....11
1.1 Benchmark Sponsor....11
1.2 Parameter Settings....11
1.3 Configuration Diagrams....11
1.3.1 Priced and Measured Configurations....12
2 Clause 1: Logical Database Design Related Items....13
2.1 Database Table Definitions....13
2.2 Database Organization....13
2.3 Horizontal Partitioning....13
2.4 Replication....13
3 Clause 2: Queries and Update Functions Related Items....14
3.1 Query Language....14
3.2 Random Number Generation....14
3.3 Substitution Parameters Generation....14
3.4 Query Text and Output Data from Database....14
3.5 Query Substitution Parameters and Seeds Used....14
3.6 Query Isolation Level....14
3.7 Refresh Function Implementation....14
4 Clause 3: Database System Properties Related Items....15
4.1 Atomicity Requirements....15
4.1.1 Atomicity of Completed Transactions....15
4.1.2 Atomicity of Aborted Transactions....15
4.2 Consistency Requirements....15
4.2.1 Consistency Condition....15
4.2.2 Consistency Tests....16
4.3 Isolation Requirements....16
4.3.1 Isolation Test 1....16
4.3.2 Isolation Test 2....16
4.3.3 Isolation Test 3....16
4.3.4 Isolation Test 4....17
4.3.5 Isolation Test 5....17
4.3.6 Isolation Test 6....17
4.4 Durability Requirements....17
4.4.1 Failure of Durable Medium Containing Recovery Log Data, and Loss of System Power/Memory... 17
4.4.2. Loss of Switch Power....18
5 Clause 4: Scaling and Database Population Related Items....19
5.1 Cardinality of Tables....19
5.2 Distribution of Tables and Logs....19
5.3 Database Partition / Replication Mapping....20
5.4 RAID Implementation....20
5.5 DBGEN Modifications....20
5.6 Database Load Time....20
5.7 Data Storage Ratio....20
5.8 Database Load Mechanism Details and Illustration....21
5.9 Qualification Database Configuration....21
6 Clause 5: Performance Metrics and Execution Rules Related Items....22
6.1 System Activity between Load and Performance Tests....22
6.2 Steps in the Power Test....22
6.3 Timing Intervals for Each Query and Refresh Function....22
6.4 Number of Streams for the Throughput Test....22
6.5 Start and End Date/Times for Each Query Stream....22
6.6 Total Elapsed Time for the Measurement Interval....22

6.7 Refresh Function Start Date/Time and Finish Date/Time 22
6.8 Timing Intervals for Each Query and Each Refresh Function for Each Stream....22
6.9 Performance Metrics....22
6.10 Performance Metric and Numerical Quantities from Both Runs....23
6.11 System Activity between Tests....23
7 Clause 6: SUT and Driver Implementation Related Items....24
7.1 Driver 24
7.2 Implementation-Specific Layer 24
7.3 Profile-Directed Optimization....24
8 Clause 7: Pricing Related Items....25
8.1 Hardware and Software Components....25
8.2 Three-Year Cost of System Configuration....25
8.4 Country-Specific Pricing....25
9 Clause 8: Audit Related Items....26
9.1 Auditor's Report....26
Appendix A: Tunables and System Configuration....27
DB2 UDB 8.2 Database Configuration....27
Node 0....27
Node 1....28
Node 2....28
Node 3....29
Node 4....30
Node 5....31
Node 6....32
Node 7....33
DB2 UDB Database Manager Configuration....34
DB2 Registry Parameters....35
bpvars.cfg....35
/etc/raw....35
/etc/sysctl.conf....35
/etc/raidtab....35
Appendix B: Database Build Scripts....36
tpcd.setup....36
buildtpcd....38
load_db2st....46
load_dbmcfg.ddl....46
load_dbcfg.ddl....46
create_nodegroups.ddl....46
change_logs.ksh....46
create_bufferpools.ddl....46
createuftbl....46
preloadUF.ksh....47
ploaduf1....47
ploaduf2....47
DeleteSampleUFData.sql....47
create_tablespaces.ddl....47
create_tables.ddl....49
load_database.csh....50
load_all.sql....50
create_indexes.ddl....51
runstats.ddl....52
chng_ovrhd.sql....52
run_db2set.ksh....52
run_dbmcfg.ddl....52
run_dbcfg.ddl....53
load_dbcfg_qual.ddl....53

change_logs_qual.ksh....53
create_tablespaces_qual.ddl....53
chng_ovrhd_qual.sql....55
run_dbcfg_qual.ddl....55
run_dbmcfg_qual.ddl....55
Appendix C: Qualification Queries....56
Qualification Query Output....56
QUERY 1....56
QUERY 2....56
QUERY 3....58
QUERY 4....58
QUERY 5....58
QUERY 6....59
QUERY 7....59
QUERY 8....59
QUERY 9....60
QUERY 11....61
QUERY 12....62
QUERY 14....63
QUERY 15....63
QUERY 16....63
QUERY 17....64
QUERY 18....64
QUERY 19....65
QUERY 20....66
QUERY 21....66
QUERY 22....67
First 10 Rows....67
Query Substitution Variables....69
Appendix D: Driver Source Code....73
setupRun....73
runpower....73
nthroughput....76
Makefile.linux....79
tpcdUF.sqc....79
tpcdbatch.h....84
tpcdbatch.sqc....85
getvars....110
macro.pl....111
tpcdmacro.pl....185
buildtpcdbatch....186
Appendix E: ACID Transaction Source Code....188
Makefile....188
acid.h....188
acid.sqc....188
mainacid.c....197
Appendix F: Price Quotes....201

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 2.1.0) August 14, 2003. This is the full disclosure report for benchmark testing of an SGI Altix 350 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 industry-wide relevance while maintaining a sufficient degree of ease of implementation. This benchmark illustrates decision support systems that:

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

TPC-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.

Silicon Graphics, Inc. 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 database parameters/options 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.

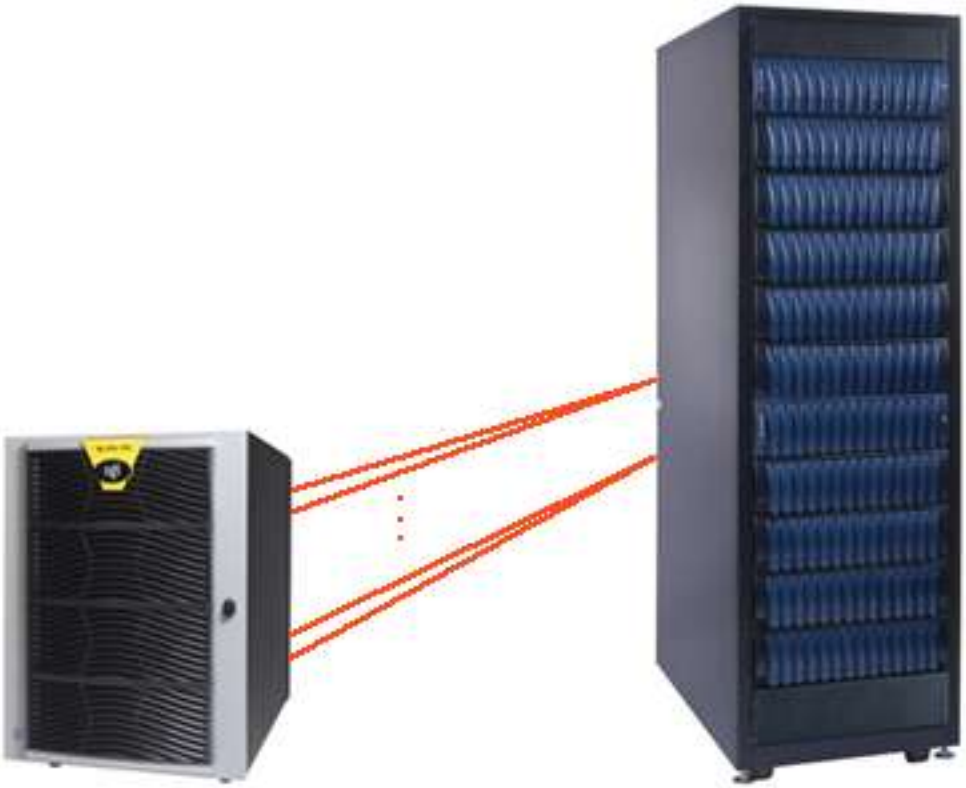
1.3 Configuration Diagrams

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, middle ware 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



SGI Altix 350
8 x 1.6GHz, 6MB-L3
64GB
2 x 73GB, 10K rpm disk drives
8 x dual-port 2Gb FC HBA (QLA 2342)

8x TP9300 base drive enclosures
68 x 73GB, 15K rpm disk drives

For full details of the priced configuration see the pricing spreadsheet in the Executive Summary.

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).

Horizontal partitioning was used for all tables except for the nation and region tables. See Appendix B, "Database Build Scripts."

2.4 Replication

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

No replication was used.

3 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.1, "Qualification Queries," contains the output for each of the 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 SGI Altix 350. 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 7 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.

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

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.

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

Failure of Database Transaction Log Disk and Loss of System Power/Memory:

1. The consistency test described in section 4.2.1 was verified.
2. The current count of the total number of records in the HISTORY table was determined giving hist1.
3. A test to run 200 ACID transactions on each of 7 execution streams was started such that each stream executes a different set of transactions.
4. One of the disks containing the DB2 transaction log recovery data was removed from a enclosure after at least 100 ACID transactions had completed from each of the execution streams.
5. Because the disks were in RAID 1 configuration the applications continued running the ACID transactions.
6. The system was shutdown by resetting from the L2 controller, after an additional 30 transactions had completed for each stream
7. The system was powered back on and rebooted.
8. All volumes were re-established and mirrored volumes were synchronized.
9. Step 2 was performed giving hist2. It was verified that hist2 - hist1 was greater than or equal to the number of records in the success file.
10. Consistency condition described in 4.2.1 was verified.

Failure of Database Data/Index/Temp Disk:

1. The current count of the total number of records in the HISTORY table was determined giving hist1.
2. A test to run 200 ACID transactions on each of 7 execution streams was started such that each stream executes a different set of transactions.
3. One of the disks containing the DB2 data/index/tempspace was power off after at least 100 ACID transactions had completed from each of the execution streams.
4. The failure was seen by the acid execution script and by the database manager.
5. The disk was restored and repartitioned.
6. The database was recovered by restoring from the previous backup and rolling forward the transaction logs.
7. Step 2 was performed giving hist2. It was verified that hist2 - hist1 was greater than or equal to the number of records in the success file.
8. Consistency condition described in 4.2.1 was verified.

4.4.2. Loss of Switch Power

No switch was used in this configuration so no this test was ommited.

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

5.2 Distribution of Tables and Logs

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

Eight disk enclosures each housing 8 x 73GB FC disk drives were used for the benchmark. Each enclosure was configured with 2 x RAID 0 LUNs (4 disk). Two of the eight disk enclosures contained an additional 2 disks configured as 1 x RAID 1 LUN (2 drive). Each of the two RAID controllers in the enclosures was attached to a port of a QLA2342 HBA. See the disk/lun mapping table for specific information.

DB2 was configured to use 8 logical nodes. Each logical node used 2 x RAID 0 LUNs for data, index, and temporary tablespace data and 1 x RAID 1 LUN partition for recovery logs. Data, index, and temporary tablespaces were spread across all RAID 0 luns, except for nation and region tables which we only on nodegroup 0. DB2 recovery logs were located on RAID 1 LUNs.

The Operating System, benchmark executing programs and the database software resided on an internal disk. An md software RAID 1 volume created using a single partition on two of the RAID 0 LUNs housed the database home directory. All flatfiles used to load the database and the RFs were stored on an lvm volume created on a single partition striped across 8 RAID 0 LUNs. Database backups were also stored on the lvm volume.

RAID 0 LUNs*	
Slice	Tablespace
1	temp_32k
2	line_data
3	line_index
5	other_data
6	other_index
7	temp_4k
8	qual_temp32
9	qual_line_data
12	small_data [sdc only]
13	Flatfiles

* sdc, sde, sdf, sdg, sdh, sdi, sdj, sdk, sdl, sdm, sdn, sdo, sdp, sdq, sdr, sdt

RAID 1 LUNs		
Slice	sdd	sds
1	node0_log	node4_log
2	node1_log	node5_log
3	node2_log	node6_log
4	node3_log	node7_log

PCI Bus: slot:port	Lun #	Dev Name	RAID Level	DB2 Node
02:01.0	0	sdc	0	0
02:01.0	2	sdd	1	0-3
02:01.1	1	sde	0	0
02:02.0	0	sdf	0	1
02:02.1	1	sdg	0	1
11:02.0	0	sdh	0	2
11:02.1	1	sdi	0	2
12:02.0	0	sdj	0	3
12:02.1	1	sdk	0	3
21:02.0	0	sdl	0	4
21:02.1	1	sdm	0	4
22:02.0	0	sdn	0	5
22:02.1	1	sdo	0	5
31:02.0	0	sdp	0	6
31:02.1	1	sdq	0	6
32:02.0	0	sdr	0	7
32:02.0	2	sds	1	4-7

5.3 Database Partition / Replication Mapping

The mapping of database partitions/replications must be explicitly described.
The database was not replicated. The database was logically partitioned into 8 logical nodes.

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 level 1 was used for all recovery logs. RAID level 0 was used across all database tables, indexes, and temporary tablespaces. Hardware RAID was used for both level 1 and level 0, and was provided by RAID controllers of the SGI TP9300 Storage Array.

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	Total Disk Space	Scale Factor	Storage Ratio
73GB 15K RPM FC Drives	68	4964GB		
73GB 10KRPM SCSI Drive	2	146GB		
Total	70	5110GB	300	17.03

The data storage ratio is 17.03, derived by dividing 5110GB by the database size of 300GB.

5.8 Database Load Mechanism Details and Illustration

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

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

The NATION and REGION tables were created on node 0 and then loaded from dbgen output. The other tables were loaded on 8 logical nodes.

The tables were loaded as depicted in Figure 4-1.

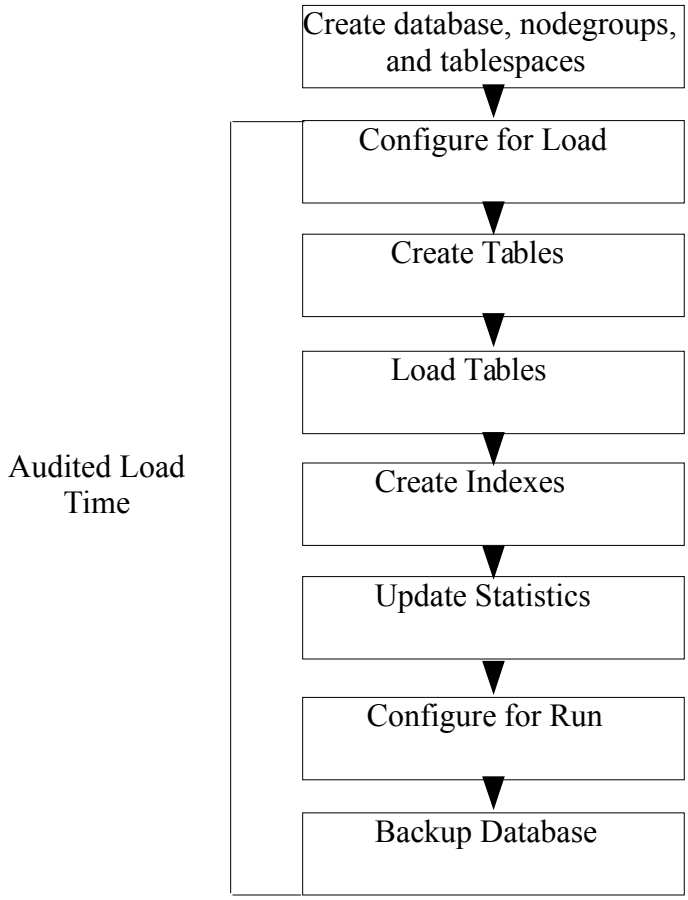


Figure 4-1. Database Load Procedure

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 same subset of disks to create and load the data. There was no difference between the configuration of the qualification database and the test database. See Section 5.2 for details.

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 @ 300GB	QthH @ 300GB	QphH @ 300GB
Run1	11,858.3	6,571.4	9,962.5
Run2	13,500.6	7,460.0	10,035.7

6.11 System Activity between Tests

Rebooted SUT between performance tests.

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.

8.3Availability 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 system as priced will be generally available as of 2005/03/25. All Hardware and System Software are currently available. The DB2 UDB V8.2 database software and FixPak 1 will be available as of 2005/03/25.

8.4 Country-Specific Pricing

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

The configuration is priced for the United States of America.

9 Clause 8: 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 Lorna Livingtree of Performance Metrics, Inc. Further information can be downloaded from www.tpc.org.

PERFORMANCE METRICS INC.
TPC Certified Auditors

March 18, 2005

Mr. Doug Nelson
Silicon Graphics, Inc.
1500 Crittenden Lane
Mountain View, CA 94043

I have verified the TPC Benchmark™ H for the following configuration:

Platform: SGI Altix 350
Database Manager: DB2 UDB 8.2
Operating System: SUSE Linux Enterprise Server

CPU's	Memory	Total Disks	Qpph@ 300GB	QthH@300GB	QphH@300GB
8 Intel IA-64 @ 1.6 Ghz	64 GB	68 @ 73GB 15K rpm 2 @ 73GB 10K rpm	**13,311.4**	**7,456.1**	**9,962.5**

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

- The database tables were defined with the proper columns, layout and sizes.
- The tested database was correctly scaled and populated for 300 GB using DBGEN. The version of DBGEN was 1.3.0.
- The qualification database layout was identical to the tested database except for the number and size of the files.
- The query text was verified to use only compliant variants.
- The executable query text was generated by QGEN and submitted to DB2 through a compliant implementation specific layer. The version of QGEN was 1.3.0.
- The validation of the query text against the qualification database produced compliant results, with the specific exceptions noted below.
- The refresh functions were properly implemented and executed the correct number of inserts and deletes.

PO Box 984 Klamath, CA 95548
(707) 482-0523 fax: (707) 482-0575

Page 1
email: LornaL@PerfMetrics.com

Appendix A: Tunables and System Configuration

DB2 UDB 8.2 Database Configuration

Node 0

```
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) = ANY
 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) = 20000
 Size of database shared memory (4KB)  (DATABASE_MEMORY) =
AUTOMATIC
 Catalog cache size (4KB)             (CATALOGCACHE_SZ) = 386
 Log buffer size (4KB)                     (LOGBUFSZ) = 2048
 Utilities heap size (4KB)              (UTIL_HEAP_SZ) = 10000
 Buffer pool size (pages)                  (BUFFPAGE) = 5000
 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) = 40000

 Max size of appl. group mem set (4KB) (APPGROUP_MEM_SZ) = 5000
 Percent of mem for appl. group heap   (GROUPHEAP_RATIO) = 70
 Max appl. control heap size (4KB)     (APP_CTL_HEAP_SZ) = 2058

 Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = 320000
 Sort list heap (4KB)                     (SORTHEAP) = 10000
 SQL statement heap (4KB)                   (STMTHEAP) = 20000
 Default application heap (4KB)           (APPLHEAPSZ) = 16000
 Package cache size (4KB)                (PCKCACHESZ) = 640
 Statistics heap size (4KB)             (STAT_HEAP_SZ) = 10000

 Interval for checking deadlock (ms)        (DLCHKTIME) = 5000
 Percent. of lock lists per application       (MAXLOCKS) = 20
 Lock timeout (sec)                      (LOCKTIMEOUT) = -1

 Changed pages threshold              (CHNGPGS_THRESH) = 60
 Number of asynchronous page cleaners   (NUM_IOCLEANERS) = 8
 Number of I/O servers                  (NUM_IOSERVERS) = 16
 Index sort flag                          (INDEXSORT) = YES
 Sequential detect flag                     (SEQDETECT) = YES
 Default prefetch size (pages)        (DFT_PREFETCH_SZ) = 16

 Track modified pages                       (TRACKMOD) = OFF

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

 Max number of active applications          (MAXAPPLS) = 40
 Average number of active applications       (AVG_APPLS) = 1
 Max DB files open per application          (MAXFILOP) = 1024

 Log file size (4KB)                     (LOGFILSIZ) = 50000
 Number of primary log files              (LOGPRIMARY) = 4
 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                                 = S0000003.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 chckpt (SOFTMAX) = 100
 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
(ACCESS)
 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


papers: db2 get database configuration for TPCD completed ok
```

Node 1

```
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) = ANY
 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) = 20000
 Size of database shared memory (4KB)  (DATABASE_MEMORY) =
AUTOMATIC
 Catalog cache size (4KB)             (CATALOGCACHE_SZ) = 386
 Log buffer size (4KB)                     (LOGBUFSZ) = 2048
 Utilities heap size (4KB)              (UTIL_HEAP_SZ) = 10000
 Buffer pool size (pages)                  (BUFFPAGE) = 5000
 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) = 40000

 Max size of appl. group mem set (4KB) (APPGROUP_MEM_SZ) = 5000
 Percent of mem for appl. group heap   (GROUPHEAP_RATIO) = 70
 Max appl. control heap size (4KB)     (APP_CTL_HEAP_SZ) = 2058

 Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = 320000
 Sort list heap (4KB)                      (SORTHEAP) = 10000
 SQL statement heap (4KB)                  (STMTHEAP) = 20000
 Default application heap (4KB)           (APPLHEAPSZ) = 16000
 Package cache size (4KB)                (PCKCACHESZ) = 640
 Statistics heap size (4KB)             (STAT_HEAP_SZ) = 10000

 Interval for checking deadlock (ms)        (DLCHKTIME) = 5000
 Percent. of lock lists per application       (MAXLOCKS) = 20
 Lock timeout (sec)                      (LOCKTIMEOUT) = -1

 Changed pages threshold               (CHNGPGS_THRESH) = 60
 Number of asynchronous page cleaners   (NUM_IOCLEANERS) = 8
 Number of I/O servers                  (NUM_IOSERVERS) = 16
 Index sort flag                         (INDEXSORT) = YES
 Sequential detect flag                  (SEQDETECT) = YES
 Default prefetch size (pages)        (DFT_PREFETCH_SZ) = 16

 Track modified pages                      (TRACKMOD) = OFF

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

 Max number of active applications           (MAXAPPLS) = 40
 Average number of active applications       (AVG_APPLS) = 1
 Max DB files open per application          (MAXFILOP) = 1024

 Log file size (4KB)                     (LOGFILSIZ) = 50000
 Number of primary log files             (LOGPRIMARY) = 4
 Number of secondary log files           (LOGSECOND) = 2
 Changed path to log files               (NEWLOGPATH) =
 Path to log files                                   = /dev/raw/raw2
 Overflow log path                   (OVERFLOWLOGPATH) =
 Mirror log path                        (MIRRORLOGPATH) =
 First active log file                               = S0000000.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 chckpt (SOFTMAX) = 100
 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
(ACCESS)
 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


papers: db2 get database configuration for TPCD completed ok
```

Node 2

```
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) = ANY
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) = 20000
Size of database shared memory (4KB)  (DATABASE_MEMORY) =
AUTOMATIC
Catalog cache size (4KB)            (CATALOGCACHE_SZ) = 386
Log buffer size (4KB)                     (LOGBUFSZ) = 2048
Utilities heap size (4KB)              (UTIL_HEAP_SZ) = 10000
Buffer pool size (pages)                  (BUFFPAGE) = 5000
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) = 40000

Max size of appl. group mem set (4KB) (APPGROUP_MEM_SZ) = 5000
Percent of mem for appl. group heap   (GROUPHEAP_RATIO) = 70
Max appl. control heap size (4KB)     (APP_CTL_HEAP_SZ) = 2058

Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = 320000
Sort list heap (4KB)                      (SORTHEAP) = 10000
SQL statement heap (4KB)                  (STMTHEAP) = 20000
Default application heap (4KB)           (APPLHEAPSZ) = 16000
Package cache size (4KB)                (PCKCACHESZ) = 640
Statistics heap size (4KB)             (STAT_HEAP_SZ) = 10000

Interval for checking deadlock (ms)        (DLCHKTIME) = 5000
Percent. of lock lists per application       (MAXLOCKS) = 20
Lock timeout (sec)                      (LOCKTIMEOUT) = -1

Changed pages threshold               (CHNGPGS_THRESH) = 60
Number of asynchronous page cleaners   (NUM_IOCLEANERS) = 8
Number of I/O servers                  (NUM_IOSERVERS) = 16
Index sort flag                          (INDEXSORT) = YES
Sequential detect flag                   (SEQDETECT) = YES
Default prefetch size (pages)        (DFT_PREFETCH_SZ) = 16

Track modified pages                      (TRACKMOD) = OFF

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

Max number of active applications           (MAXAPPLS) = 40
Average number of active applications        (AVG_APPLS) = 1
Max DB files open per application           (MAXFILOP) = 1024

Log file size (4KB)                     (LOGFILSIZ) = 50000
Number of primary log files             (LOGPRIMARY) = 4
Number of secondary log files            (LOGSECOND) = 2

Changed path to log files                (NEWLOGPATH) =
Path to log files                           = /dev/raw/raw3
Overflow log path                   (OVERFLOWLOGPATH) =
Mirror log path                     (MIRRORLOGPATH) =
First active log file                       = S0000000.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 chckpt (SOFTMAX) = 100
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
(ACCESS)
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
```

Node 3

papers: db2 get database configuration for TPCD completed ok

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) = ANY
```

```
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) = 20000
Size of database shared memory (4KB)  (DATABASE_MEMORY) =
AUTOMATIC
Catalog cache size (4KB)             (CATALOGCACHE_SZ) = 386
Log buffer size (4KB)                      (LOGBUFSZ) = 2048
Utilities heap size (4KB)               (UTIL_HEAP_SZ) = 10000
Buffer pool size (pages)                   (BUFFPAGE) = 5000
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) = 40000

Max size of appl. group mem set (4KB) (APPGROUP_MEM_SZ) = 5000
Percent of mem for appl. group heap   (GROUPHEAP_RATIO) = 70
Max appl. control heap size (4KB)     (APP_CTL_HEAP_SZ) = 2058

Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = 320000
Sort list heap (4KB)                       (SORTHEAP) = 10000
SQL statement heap (4KB)                   (STMTHEAP) = 20000
Default application heap (4KB)           (APPLHEAPSZ) = 16000
Package cache size (4KB)                (PCKCACHESZ) = 640
Statistics heap size (4KB)              (STAT_HEAP_SZ) = 10000

Interval for checking deadlock (ms)        (DLCHKTIME) = 5000
Percent. of lock lists per application       (MAXLOCKS) = 20
Lock timeout (sec)                       (LOCKTIMEOUT) = -1

Changed pages threshold                (CHNGPGS_THRESH) = 60
Number of asynchronous page cleaners   (NUM_IOCLEANERS) = 8
Number of I/O servers                  (NUM_IOSERVERS) = 16
Index sort flag                          (INDEXSORT) = YES
Sequential detect flag                   (SEQDETECT) = YES
Default prefetch size (pages)        (DFT_PREFETCH_SZ) = 16

Track modified pages                       (TRACKMOD) = OFF

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

Max number of active applications          (MAXAPPLS) = 40
Average number of active applications       (AVG_APPLS) = 1
Max DB files open per application          (MAXFILOP) = 1024

Log file size (4KB)                      (LOGFILSIZ) = 50000
Number of primary log files                (LOGPRIMARY) = 4
Number of secondary log files              (LOGSECOND) = 2
Changed path to log files                (NEWLOGPATH) =
Path to log files                                = /dev/raw/raw4
Overflow log path                     (OVERFLOWLOGPATH) =
Mirror log path                         (MIRRORLOGPATH) =
First active log file                            = S0000000.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 chckpt (SOFTMAX) = 100
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
(ACCESS)
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
```

papers: db2 get database configuration for TPCD completed ok

Node 4

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) = ANY
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) = 20000
 Size of database shared memory (4KB)  (DATABASE_MEMORY) =
AUTOMATIC
 Catalog cache size (4KB)            (CATALOGCACHE_SZ) = 386
 Log buffer size (4KB)                    (LOGBUFSZ) = 2048
 Utilities heap size (4KB)              (UTIL_HEAP_SZ) = 10000
 Buffer pool size (pages)                  (BUFFPAGE) = 5000
 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) = 40000

 Max size of appl. group mem set (4KB) (APPGROUP_MEM_SZ) = 5000
 Percent of mem for appl. group heap   (GROUPHEAP_RATIO) = 70
 Max appl. control heap size (4KB)     (APP_CTL_HEAP_SZ) = 2058

 Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = 320000
 Sort list heap (4KB)                      (SORTHEAP) = 10000
 SQL statement heap (4KB)                  (STMTHEAP) = 20000
 Default application heap (4KB)           (APPLHEAPSZ) = 16000
 Package cache size (4KB)                (PCKCACHESZ) = 640
 Statistics heap size (4KB)            (STAT_HEAP_SZ) = 10000

 Interval for checking deadlock (ms)        (DLCHKTIME) = 5000
 Percent. of lock lists per application       (MAXLOCKS) = 20
 Lock timeout (sec)                      (LOCKTIMEOUT) = -1

 Changed pages threshold              (CHNGPGS_THRESH) = 60
 Number of asynchronous page cleaners   (NUM_IOCLEANERS) = 8
 Number of I/O servers                 (NUM_IOSERVERS) = 16
 Index sort flag                         (INDEXSORT) = YES
 Sequential detect flag                  (SEQDETECT) = YES
 Default prefetch size (pages)        (DFT_PREFETCH_SZ) = 16

 Track modified pages                      (TRACKMOD) = OFF

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

 Max number of active applications           (MAXAPPLS) = 40
 Average number of active applications       (AVG_APPLS) = 1
 Max DB files open per application           (MAXFILOP) = 1024

 Log file size (4KB)                      (LOGFILSIZ) = 50000
 Number of primary log files              (LOGPRIMARY) = 4
 Number of secondary log files            (LOGSECOND) = 2
 Changed path to log files               (NEWLOGPATH) =
 Path to log files                         = /dev/raw/raw5
 Overflow log path                   (OVERFLOWLOGPATH) =
 Mirror log path                     (MIRRORLOGPATH) =
 First active log file                     = S0000000.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 chckpt (SOFTMAX) = 100
 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
(ACCESS)
 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
```

papers: db2 get database configuration for TPCD completed ok

Node 5

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) = ANY
 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) = 20000
Size of database shared memory (4KB)  (DATABASE_MEMORY) =
AUTOMATIC
Catalog cache size (4KB)            (CATALOGCACHE_SZ) = 386
Log buffer size (4KB)                       (LOGBUFSZ) = 2048
Utilities heap size (4KB)             (UTIL_HEAP_SZ) = 10000
Buffer pool size (pages)                    (BUFFPAGE) = 5000
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) = 40000

Max size of appl. group mem set (4KB) (APPGROUP_MEM_SZ) = 5000
Percent of mem for appl. group heap   (GROUPHEAP_RATIO) = 70
Max appl. control heap size (4KB)     (APP_CTL_HEAP_SZ) = 2058

Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = 320000
Sort list heap (4KB)                        (SORTHEAP) = 10000
SQL statement heap (4KB)                    (STMTHEAP) = 20000
Default application heap (4KB)              (APPLHEAPSZ) = 16000
Package cache size (4KB)                    (PCKCACHESZ) = 640
Statistics heap size (4KB)            (STAT_HEAP_SZ) = 10000

Interval for checking deadlock (ms)         (DLCHKTIME) = 5000
Percent. of lock lists per application      (MAXLOCKS) = 20
Lock timeout (sec)                          (LOCKTIMEOUT) = -1

Changed pages threshold               (CHNGPGS_THRESH) = 60
Number of asynchronous page cleaners   (NUM_IOCLEANERS) = 8
Number of I/O servers                 (NUM_IOSERVERS) = 16
Index sort flag                       (INDEXSORT) = YES
Sequential detect flag                (SEQDETECT) = YES
Default prefetch size (pages)         (DFT_PREFETCH_SZ) = 16

Track modified pages                        (TRACKMOD) = OFF

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

Max number of active applications           (MAXAPPLS) = 40
Average number of active applications       (AVG_APPLS) = 1
Max DB files open per application           (MAXFILOP) = 1024

Log file size (4KB)                         (LOGFILSIZ) = 50000
Number of primary log files                 (LOGPRIMARY) = 4
Number of secondary log files               (LOGSECOND) = 2
Changed path to log files                   (NEWLOGPATH) =
Path to log files                           = /dev/raw/raw6
Overflow log path                           (OVERFLOWLOGPATH) =
Mirror log path                             (MIRRORLOGPATH) =
First active log file                       = S0000000.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 chckpt (SOFTMAX) = 100
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
(ACCESS)
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
```

papers: db2 get database configuration for TPCD completed ok

Node 6

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) = ANY
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) = 20000
Size of database shared memory (4KB)  (DATABASE_MEMORY) =
AUTOMATIC
Catalog cache size (4KB)            (CATALOGCACHE_SZ) = 386
Log buffer size (4KB)                       (LOGBUFSZ) = 2048
Utilities heap size (4KB)             (UTIL_HEAP_SZ) = 10000
Buffer pool size (pages)                    (BUFFPAGE) = 5000
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) = 40000

Max size of appl. group mem set (4KB) (APPGROUP_MEM_SZ) = 5000
Percent of mem for appl. group heap   (GROUPHEAP_RATIO) = 70
Max appl. control heap size (4KB)     (APP_CTL_HEAP_SZ) = 2058

Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = 320000
Sort list heap (4KB)                       (SORTHEAP) = 10000
SQL statement heap (4KB)                   (STMTHEAP) = 20000
Default application heap (4KB)             (APPLHEAPSZ) = 16000
Package cache size (4KB)                 (PCKCACHESZ) = 640
Statistics heap size (4KB)               (STAT_HEAP_SZ) = 10000

Interval for checking deadlock (ms)        (DLCHKTIME) = 5000
Percent. of lock lists per application       (MAXLOCKS) = 20
Lock timeout (sec)                       (LOCKTIMEOUT) = -1

Changed pages threshold                (CHNGPGS_THRESH) = 60
Number of asynchronous page cleaners   (NUM_IOCLEANERS) = 8
Number of I/O servers                  (NUM_IOSERVERS) = 16
Index sort flag                        (INDEXSORT) = YES
Sequential detect flag                 (SEQDETECT) = YES
Default prefetch size (pages)          (DFT_PREFETCH_SZ) = 16

Track modified pages                       (TRACKMOD) = OFF

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

Max number of active applications          (MAXAPPLS) = 40
Average number of active applications      (AVG_APPLS) = 1
Max DB files open per application          (MAXFILOP) = 1024

Log file size (4KB)                      (LOGFILSIZ) = 50000
Number of primary log files              (LOGPRIMARY) = 4
Number of secondary log files            (LOGSECOND) = 2
Changed path to log files                (NEWLOGPATH) =
Path to log files                                  = /dev/raw/raw7
Overflow log path                        (OVERFLOWLOGPATH) =
Mirror log path                          (MIRRORLOGPATH) =
First active log file                              = S0000000.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 chckpt (SOFTMAX) = 100
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
(ACCESS)
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
```

papers: db2 get database configuration for TPCD completed ok

Node 7

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) = ANY
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) = 20000
Size of database shared memory (4KB)  (DATABASE_MEMORY) =
AUTOMATIC
Catalog cache size (4KB)                    (CATALOGCACHE_SZ) = 386
Log buffer size (4KB)                       (LOGBUFSZ) = 2048
Utilities heap size (4KB)                   (UTIL_HEAP_SZ) = 10000
Buffer pool size (pages)                    (BUFFPAGE) = 5000
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) = 40000

Max size of appl. group mem set (4KB) (APPGROUP_MEM_SZ) = 5000
Percent of mem for appl. group heap   (GROUPHEAP_RATIO) = 70
Max appl. control heap size (4KB)     (APP_CTL_HEAP_SZ) = 2058

Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = 320000
Sort list heap (4KB)                       (SORTHEAP) = 10000
SQL statement heap (4KB)                   (STMTHEAP) = 20000
Default application heap (4KB)             (APPLHEAPSZ) = 16000
Package cache size (4KB)                 (PCKCACHESZ) = 640
```

```
 Statistics heap size (4KB)              (STAT_HEAP_SZ) = 10000

 Interval for checking deadlock (ms)        (DLCHKTIME) = 5000
 Percent. of lock lists per application       (MAXLOCKS) = 20
 Lock timeout (sec)                      (LOCKTIMEOUT) = -1

 Changed pages threshold              (CHNGPGS_THRESH) = 60
 Number of asynchronous page cleaners   (NUM_IOCLEANERS) = 8
 Number of I/O servers                  (NUM_IOSERVERS) = 16
 Index sort flag                         (INDEXSORT) = YES
 Sequential detect flag                   (SEQDETECT) = YES
 Default prefetch size (pages)        (DFT_PREFETCH_SZ) = 16

 Track modified pages                     (TRACKMOD) = OFF

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

 Max number of active applications          (MAXAPPLS) = 40
 Average number of active applications       (AVG_APPLS) = 1
 Max DB files open per application          (MAXFILOP) = 1024

 Log file size (4KB)                     (LOGFILSIZ) = 50000
 Number of primary log files              (LOGPRIMARY) = 4
 Number of secondary log files             (LOGSECOND) = 2
 Changed path to log files               (NEWLOGPATH) =
 Path to log files                                  = /dev/raw/raw8
 Overflow log path                    (OVERFLOWLOGPATH) =
 Mirror log path                        (MIRRORLOGPATH) =
 First active log file                              = S0000000.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 chckpt (SOFTMAX) = 100
 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
(ACCESS)
 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

papers: db2 get database configuration for TPCD completed ok
```

DB2 UDB Database Manager Configuration

```
   Node type = Enterprise Server Edition with local and remote clients

 Database manager configuration release level            = 0x0a00

 CPU speed (millisec/instruction)             (CPUSPEED) = 2.000000e-06
 Communications bandwidth (MB/sec)      (COMM_BANDWIDTH) =
1.000000e+01

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

 Default charge-back account          (DFT_ACCOUNT_STR) =

 Java Development Kit installation path       (JDK_PATH) = /opt/IBMJava2-
142

 Diagnostic error capture level             (DIAGLEVEL) = 3
 Notify Level                          (NOTIFYLEVEL) = 3
 Diagnostic data directory path             (DIAGPATH) =

 Default database monitor switches
   Buffer pool                       (DFT_MON_BUFPOOL) = OFF
   Lock                               (DFT_MON_LOCK) = OFF
   Sort                              (DFT_MON_SORT) = OFF
   Statement                          (DFT_MON_STMT) = OFF
   Table                            (DFT_MON_TABLE) = OFF
   Timestamp                     (DFT_MON_TIMESTAMP) = OFF
   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) = /mnt/disk2/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) = 140000

 Directory cache support                   (DIR_CACHE) = YES

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

 Workload impact by throttled utilities(UTIL_IMPACT_LIM) = 10

 Priority of agents                       (AGENTPRI) = SYSTEM
 Max number of existing agents             (MAXAGENTS) = 200
```

```
 Agent pool size                    (NUM_POOLAGENTS) = 64
 Initial number of agents in pool       (NUM_INITAGENTS) = 4
 Max number of coordinating agents     (MAX_COORDAGENTS) =
(MAXAGENTS - NUM_INITAGENTS)
 Max no. of concurrent coordinating agents  (MAXCAGENTS) =
MAX_COORDAGENTS
 Max number of 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) = ACCESS

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

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

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

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

 No. of int. communication buffers(4KB)(FCM_NUM_BUFFERS) = 8192
 Number of FCM request blocks              (FCM_NUM_RQB) = 4096
 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 Registry Parameters

```
DB2LINUXAIO=TRUE
DB2_LARGE_PAGE_MEM=DB
DB2_SCATTERED_IO=Y
DB2_EXTENDED_OPTIMIZATION=Y
DB2_ANTIJOIN=Y
DB2_STRIPED_CONTAINERS=ON
DB2BPVARS=/mnt/disk2/tpch/tpcd/samples/custom.linux/bpvars.cfg
DB2OPTIONS=-t -v +c
DB2COMM=tcpip
DB2CODEPAGE=819
DB2_PARALLEL_IO=*
```

---

bpvars.cfg

```
NUMPREFETCHQUEUES=4
PREFETCHQUEUESIZE=800
```

/etc/raw

```
# /etc/raw
#
# sample configuration to bind raw devices
# to block devices
#
# The format of this file is:
# raw<N>:<blockdev>
#
# example:
# ---------
# raw1:hdb1
#
# this means: bind /dev/raw/raw1 to /dev/hdb1
#
# ...
raw1:sdd1
raw2:sdd2
raw3:sdd3
raw4:sdd5
raw5:sds1
raw6:sds2
raw7:sds3
raw8:sds5
raw9:sdd6
raw10:sdd7
raw11:sdd8
raw12:sdd9
raw13:sds6
raw14:sds7
raw15:sds8
raw16:sds9
```

/etc/sysctl.conf

```
#
# add values for db2 bench
#
kernel.sem =  250 1024000 400 4096
kernel.shmall =  15728640
kernel.shmmax =  59055800320
kernel.msgmnb =  32768
vm.nr_hugepages = 224
```

/etc/raidtab

```
#
# mounted on /mnt/dbhome
raiddev /dev/md0
        raid-level              1
        nr-raid-disks           2
        nr-spare-disks          0
        chunk-size              64
        persistent-superblock   1
        device                  /dev/sdf12
        raid-disk               0
        device                  /dev/sdp12
        raid-disk               1
```

Appendix B: Database Build Scripts

---

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=/mnt/disk2/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=mln                       # uni/smp/mln/mpp
TPCD_PHYS_NODE=1                    # number of physical nodes
TPCD_LN_PER_PN=8                    # number of logical nodes per physical
node
TPCD_SF=300                         # size of the database (1=1GB,...) to
                          # get test size databases use:
                          #     0.012 = 12MB
                          #     0.1   = 100MB
TPCD_BUILD_STAGE=ALL                # where to start the build -
currently the
                          # following is possible:
                          #  ALL - do everything (create,load,
                          #       index,stats,config) (Default)
                          #  CRTTBSP - start after create db and
                          #     config setting.  Start 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=/mnt/dbhome/tpcd        # path for database (defaults to
home)
TPCD_DDLPATH=/mnt/disk2/tpch/tpcd/samples/custom.linux   # path for all
ddl files and customized
                          # scripts (load script), config files,etc
TPCD_BUFFERPOOL_DEF=create_bufferpools.ddl  # name of file with
bufferpool definitions
                          # and sizes
TPCD_NODEGROUP_DEF=create_nodegroups.ddl   # name of file in
ddlpath with nodegroup
                          # definitions
TPCD_EXPLAIN_DDL=NULL               #  file with DDL for explains
statments
                          # if this is NULL then uses the default
                          # and puts it in USERSPACE1 across all
                          # nodes...nt 1TB found it was faster if
                          # just in a single node nodegroup
TPCD_TBSP_DDL=create_tablespaces.ddl   # ddl file for tablespaces
TPCD_DDL=create_tables.ddl          # ddl file for tables
TPCD_QUAL_TBSP_DDL=create_tablespaces_qual.ddl # ddl file for
tablespaces for qual
TPCD_QUAL_DDL=create_tables.ddl     # ddl file for qualification database
                          # tablespaces and tables should be identical
                          # to regular ddl except container names
TPCD_INDEXDDL=create_indexes.ddl    # 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=runstats.ddl          # 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=/mnt/disk2/tpch/tpcd/appendix_latest/dbgen     # path name
to data generation code

                          # Parameters used to specify source of
                          # data for load scripts
TPCD_INPUT=/flatfiles/              # NULL - use dbgen generated data  OR
                          # path name -  to the pre-generated
                          #           flat files
                          #
TPCD_QUAL_INPUT=/flatfiles_1/qual_flatfiles # NULL - use dbgen
generated data  OR
                          # path name -  to the pre-generated
                          #           flat files

TPCD_TAILOR_DIR=/mnt/disk2/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=load_db2st     # Script that contains the
db2set commands
                          # for the load process Use NULL if not
                          # specified
TPCD_LOAD_CONFIGFILE=load_dbcfg.ddl       #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=load_dbmcfg.ddl   # 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=load_dbcfg_qual.ddl          # 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=load_dbmcfg.ddl      # 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
                          # used in load script only
```

```
TPCD_SORTBUF=4096                    # sortbuf size for LOAD
                                     # used in load script only
TPCD_LOAD_PARALLELISM=0              # degree of parallelism to use on load
                                     # 0 = use the "intelligent default" that
                                     # the load will chose at run time
                                     # used in load script only
TPCD_COPY_DIR=NULL                   # directory where copy image is created
                                     # on load command CURRENTLY UNUSED
                                     # used in load script only
TPCD_FASTPARSE=yes                   # use fastparse on load
                                     # used in load script only

                                     # Backup and logfile specific parameters follow:
#TPCD_BACKUP_DIR=/dev/null           # directory where backup files are placed
TPCD_BACKUP_DIR=/flatfiles_1/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=/flatfiles/db2_logfiles  # directory where log files stored..
TPCD_LOG_DIR=                        # directory where log files stored..
                                     # NULL leaves them in the dbpath
TPCD_LOG_QUAL_DIR=/flatfiles/db2_qual_logfiles  # 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=run_db2set.ksh    # Script that contains the db2set commands
                                     # for the benchmark run.  Use NULL if not
                                     # specified
TPCD_CONFIGFILE=run_dbcfg.ddl        # name of configuration file in ddl path
                                     # that will be used for the benchmark run
TPCD_DBM_CONFIG=run_dbmcfg.ddl       # name of config file for database manager
                                     # cfg parms
TPCD_QUALCONFIGFILE=run_dbcfg_qual.ddl  # name of database cfg file in ddl path
                                     # for qualification database
TPCD_DBM_QUALCONFIG=run_dbmcfg_qual.ddl  # name of config file for database
                                     # manager cfg parms

TPCD_MACHINE=small                   # set to NULL if using load config file
                                     # big/medium/small  size of machine used to
                                     # determine buffpage, sortheap,sheapthres
                                     # and ioservers parms for load, create
                                     # index and runstats
                                     # NOTE that this parameter is ignored if
                                     # a TPCD_LOAD_CONFIGFILE is specified

TPCD_SMPDEGREE=1                     # 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=/var/tmp/tpch          # place to put temp working files
TPCD_TMP_DIR=/mnt/disk2/tpch/tmp     # place to put temp working files


TPCD_SHARED_TEMP_FULL_PATHNAME=/mnt/disk2/tpch/sqllib/tmp # 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=6                     # number of streams for the throughput test
TPCD_FLATFILES=/flatfiles_1/ufdata   # where to generate/read flat files
                                     # for update functions
TPCD_STAGING_TABLE_DDL=createuftbl  # script that contains the ddl for creating
                                     # the staging tables if they are used for
                                     # the update functions
TPCD_PRELOAD_STAGING_TABLE_SCRIPT=preloadUF.ksh  # 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=DeleteSampleUFData.sql  # file that contains the sql for deleting
                                     # the preloaded data from the staging
                                     # tables
TPCD_UPDATE_IMPORT=false             # true = use import for the staging tables
                                     #   for UNI/SMP mode only (code change in
                                     #   tpcdbatch) (if not uni mode then must
                                     #   change load_update)
                                     # false = use load for staging tables
                                     # The default is false if not set.
                                     # NOTE that this parm is only for UNI/SMP
                                     # it is not for multi node invocation

TPCD_SPLIT_UPDATES=128               # number of chunks to split the update
                                     # function into.
TPCD_CONCURRENT_INSERTS=16           # number of insert chunks that are run
                                     # concurrently.  TPCD_SPLIT_UPDATES
                                     # should be evenly divisible by this number
TPCD_CONCURRENT_INSERTS_LOAD=8       # 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=128               # number of portions to split the delete
                                     # function into.
                                     # this variable is only valid in UNI/SMP
                                     # mode.
TPCD_CONCURRENT_DELETES=16           # number of delete chunks that are run
                                     # concurrently.  This number should be
                                     # evently divisible by TPCD_SPLIT_DELETES
TPCD_GEN_UPDATEPAIRS=32              # 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=5                # timing interval for vmstats/iostats
TPCD_STATS_THRU_INT=60               # timing interval for vmstats/iostats
for
                                # throughput run
TPCD_GATHER_STATS=off                # on/off   - only implement for AIX
yet
                                # on = gather statistics around power
                                #      test run  (vmstat,iostat,netstat)
                                # off = no stats gathered during power run

TPCD_UFTEMP=DATA_INDEX               # base name of tablespace(s)
where the
                                # staging tables for the update functions
                                # are created
                                # this name will be used as the
                                # basename for the tablespaces...eg
                                # UFTEMP1 UFTEMP2 ....
TPCD_HAVECOMPILER=yes                # rebuild tpcdbatch executable
                                # yes/no
TPCD_SLEEP=5                         # ?
TPCD_INLISTMAX=default               # max num of keys to delete at a time
                                # for UF2, use "default" for default.
TPCD_LOAD_SCRIPT=load/load_database.csh # script to run for loading
tables
                                # in TPCD_DDLPATH directory under mln/mpp
                                # leave as NULL if using default genloaduni
TPCD_LOAD_SCRIPT_QUAL=load/qual_load/load_database_qual.csh  #
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=/mnt/disk2/tpch/sqllib/db2dump/    # directory wehre the
db2diag.log can
                                # be found for the durability tests
TPCD_APPEND_ON=no                    # set to no if the cluster indexes are
used
TPCD_LOG_DIR_SETUP_SCRIPT=change_logs.ksh  # log changing script
TPCD_QUAL_LOG_DIR_SETUP_SCRIPT=change_logs_qual.ksh  # log
changing script
```

---

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
#-----------------------------------------------------------------------#
# 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 "getvars";
require "macro.pl";
require "tpcdmacro.pl";
require "version";

# Make output unbuffered.
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_QUAL_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){
   $ENV{"TPCD_INDEXDDL"} = "dss.index.H";
}
if (length($ENV{"TPCD_RUNSTATS"}) <= 0){
   $ENV{"TPCD_RUNSTATS"} = "dss.runstats";
}
if (length($ENV{"TPCD_CONFIGFILE"}) <= 0){
   $ENV{"TPCD_CONFIGFILE"} = "dbcfg_for_run";
}


#-----------------------------------------------------------------------#
#  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"}) <= 1){
   $ENV{"TPCD_TEMP"} = "/u/$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"};
$ddlpath=               $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_DB2SET_SCRIPT"};
$user=                          $ENV{"USER"};
$logDirScript=                  $ENV
{"TPCD_LOG_DIR_SETUP_SCRIPT"};
$quallogDirScript=      $ENV{"TPCD_QUAL_LOG_DIR_SETUP_SCRIPT"};
$logprimary=                    $ENV{"TPCD_LOGPRIMARY"};
$logsecond=                     $ENV{"TPCD_LOGSECOND"};
$logfilsiz=             $ENV{"TPCD_LOGFILSIZ"};
$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_ln="once";
  $all_pn="once";
  $once="once";
}
else{
```

```
  $all_ln="all_ln";
  $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
      $rc=&dodb_noconn("db2set DB2OPTIONS=\"  -t -v +c\";db2set
DB2NTNOCACHE=ON",$all_ln);
   }
   else{
      $rc=&dodb_noconn("db2set DB2OPTIONS=\\\"  -t -v +c\\\";db2set
DB2NTNOCACHE=ON",$all_ln);
   }
}
else{
   if (($mode eq "uni") || ($mode eq "smp")){
      $rc=&dodb_noconn("db2set DB2OPTIONS=\"-t -v +c\"",$all_ln);
   }
   else{
      $rc=&dodb_noconn("db2set DB2OPTIONS=\\\"-t -v +c\\\"",$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" )){
         $rc=system("${ddlpath}${delim}$loadsetScript");
      }
      else{
         $rc=system(" rah \" cd ${ddlpath} & $loadsetScript\" ");
      }
   }
   else{
      $rc=system("${ddlpath}${delim}$loadsetScript");
   }
   # doug ($rc == 0) || die "failure loading db2set parms from $loadsetScript
\n";
}

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

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

$rc = &setLoadConfig;

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

if( $buildStage eq "ALL" ||
   $buildStage eq "CRTTBSP" ||
   ($buildStage eq "INDEX" && $earlyindex eq "yes")){
            !&createNodegroups ||  print "ERROR:[$0] create nodegroups
failed.\n";
            !&createBufferPools || print "ERROR:[$0] create bufferpools
failed.\n";
            &outtime("*** Start of audited Load Time - starting to create
tables");
            !&createTablespaces || print "WARNING:[$0] create tablespaces
error.\n";
            !&createExplainTbls ||  print "ERROR:[$0] create EXPLAIN tables
failed.\n";
            !&createTables || print "ERROR:[$0] create tables failed.\n";

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

            # if earlyindex requested, create indexes
            if ( $earlyindex eq "yes" ){
                        !&createIndexes("early") || die "ERROR:[$0] create early
indexes failed.\n";
            }
            # start the dbgen and load.....call the specific mode for loading
(uni,smp,mln)
            !&loadData || die "ERROR:[$0] 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:[$0] create
indexes failed.\n";
            }
            if ( $extraindex ne "no" ){
                        !&createIndexes("extra") || die "ERROR:[$0] 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 specifed 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 ( $addRI ne "NULL" ){
   &outtime("*** Adding RI contraints started");
   &dodb2file($dbname,"$ddlpath${delim}$addRI",$once);
   &outtime("*** Adding RI contraints 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 buildptpcdbatch 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 ( $RealAudit eq "yes" ){
    #
        # doug comment this out to keep increasing run numbers
    #
    # &rm("$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 ( $qual eq "QUAL" ){
                if ($quallogDirScript ne "NULL"){
                        system ("$ddlpath${delim}
$quallogDirScript");
                }
                elsif ( $logDir ne "NULL" ){
                    &dodb_noconn("db2 update database configuration
for $dbname using newlogpath $logDir",$all_ln);
                }
        }
        else {
                if ($logDirScript ne "NULL"){
                        #system ("perl $ddlpath${delim}
$logDirScript");
                        system ("$ddlpath${delim}$logDirScript");
                }
                elsif ( $logDir ne "NULL" ){
                        &dodb_noconn("db2 update database
configuration for $dbname using newlogpath $logDir",$all_ln);
                }
        }
        $setLogs=0;
        $setLogString="";
        if ( $logprimary ne "NULL" ){
                $setLogString.="db2 update db cfg for $dbname using
logprimary $logprimary";
                $setLogs=1;
        }
        if ( $logsecond ne "NULL" ){
                if ( $setLogs != 0 ){
                   $setLogString.=" $sep ";
                }
                $setLogString.="db2 update db cfg for $dbname using
logsecond $logsecond";
                $setLogs=1;
        }
        if ( $logfilsiz ne "NULL" ){
                if ( $setLogs != 0 ){
                   $setLogString.=" $sep ";
                }
                $setLogString.="db2 update db cfg for $dbname using
```

```
logfilsiz $logfilsiz";
                        $setLogs=1;
            }
            if ( $setLogs != 0 ){
                        $setLogString.=" $sep ";
            }
            $setLogString.="db2 update db cfg for $dbname using logbufsz
128";
            $rc = &dodb_noconn("$setLogString",$all_ln);

}

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

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

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

            if ( $explainDDL ne "NULL" ){
                        $explnPathFile="$explainDDL";
            }
            else{
                        if ( $platform eq "ptx" ){
                                    $home=$ENV{"HOME"};
                                    $sqlpath="$home${delim}sqllib";
                        }
                        if ( $platform ne "nt" ){
                                    $home=$ENV{"HOME"};
                                    $sqlpath="$home${delim}sqllib";
                        }
                        else{
                                    $sqlpath=$ENV{"DB2PATH"};
                        }
                        $explnPathFile="$sqlpath${delim}misc${delim}
EXPLAIN.DDL";
            }
            $rc = &dodb_conn($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 $rc;
            &outtime("*** Creating the bufferpools");
            if ( $buffpooldef ne "NULL" ){
                        #run the create bufferpool ddl
                        $rc = &dodb2file($dbname,"$ddlpath${delim}
$buffpooldef",$once);
            }
}

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

#-----------------------------------------------------------------------#
# Function: createTables                                                #
#-----------------------------------------------------------------------#
sub createTables{
            my $rc;
            $rc = &dodb2file($dbname,"$ddlpath${delim}$ddl",$once);
            ($rc == 0) || return $rc;
            # update the locksize on the non-updated tables to be table level
locking
            # update the tables for appendmode
            if ($appendOn eq "yes"){
                        $rc = &dodb_conn($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 ",
                        $once);
            }
            else{
                        $rc = &dodb_conn($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 $rc;
                &outtime("*** Starting to create $indexType indexes");
                if( $indexType eq "extra"){
                        $rc = &dodb2file($dbname,"$ddlpath${delim}
$extraindex",$once);
                }elsif ($indexType eq "early" || $indexType eq "normal"){
                        $rc = &dodb2file($dbname,"$ddlpath${delim}
$indexddl",$once);
                }
                &outtime("*** Create $indexType index completed");
                return $rc;
}



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

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

        if($loadDBMconfig ne "NULL"){
                        $rc = &dodb2file_noconn("$ddlpath${delim}
$loadDBMconfig",$once);
        }
        else{
                $rc = &dodb_noconn("db2 update dbm cfg using sheapthres
$sheapthres",$once);
        }

                if (($loadconfigfile eq "") || ($loadconfigfile eq "NULL")){
                        if ( $machine eq "small" ){
                                $buffpage = 5000;
                                $sortheap = 3000;
                                $sheapthres = 8000;
                                $util_heap_sz = 5000;
                                $ioservers = 6;
                        }
                        elsif ( $machine eq "medium" ){
                                $buffpage = 10000;
                                $sortheap =  8000;
                                $sheapthres = 20000;
                                $util_heap_sz = 10000;
                                $ioservers = 10;
                        }
                        elsif ( $machine eq "big" ){
                                $buffpage = 30000;
                                $sortheap = 20000;
                                $sheapthres = 50000;
                                $util_heap_sz = 30000;
                                $ioservers = 20;
                        }
                        else {
                                die "Neither a LOAD config filename nor a
valid machine size has \
                                been specified!\n";
                        }
                        $rc = &dodb_noconn("db2 update db cfg for $dbname
using buffpage $buffpage $sep \
                        db2 update db cfg for $dbname using sortheap $sortheap
$sep \
                        db2 update db cfg for $dbname using num_iocleaners
$ioclnrs $sep \
                                db2 update db cfg for $dbname using num_ioservers
$ioservers $sep \
                                db2 update db cfg for $dbname using util_heap_sz
$util_heap_sz $sep \
                                db2 update db cfg for $dbname using chngpgs_thresh
$chngpgs",$all_ln);
                }
                else{
                                $rc = &dodb2file_noconn("$ddlpath${delim}
$loadconfigfile",$all_ln);
                }
                #($rc == 0) || return $rc;
                #if (($loadDBMconfig ne "") || ($loadDBMconfig ne "NULL")){
                                #$rc = &dodb2file_noconn("$ddlpath${delim}
$loadDBMconfig",$once);
                #}
                #else{
                                #$rc = &dodb_noconn("db2 update dbm cfg using
sheapthres $sheapthres",$once);
                #}
                ($rc == 0) || return $rc;
                &dodb_noconn("db2 terminate",$once);
                $rc = &stopStart;
                return $rc;
}
#-----------------------------------------------------------------------#
# Function: loadData                                                    #
#-----------------------------------------------------------------------#
sub loadData{
                # start the dbgen and load.....call the specific mode for loading
(uni,smp,mln)
                my $rc;
                if (( $mode eq "uni" ) || ( $mode eq "smp" )){
                                &outtime("*** Starting the load");
                                # call the appropriate dbgen/load for uni/smp
                                if ( $loadscript eq "NULL"){
                                        $rc = system("perl genloaduni $qual");
                                        ($rc == 0) || print "ERROR:[$0] genloaduni
failed rc = $rc\n";
                                }
                                else{
                                        $rc = &dodb2file_noconn("$ddlpath${delim}
$loadscript",$once);
                                        ($rc == 0) || print "ERROR:[$0] load script:
$loadscript failed. rc = $rc\n";
                                }
                }
                elsif ( $mode eq "mln" ) {
                                &outtime("*** Starting the load");
                                # call the appropriate dbgen/split/(sort)/load for mln
                                if ( $loadscript eq "NULL"){
                                        $rc = system("perl genloadmln $qual");
                                        ($rc == 0) || print "ERROR:[$0] genloadmln
failed. rc = $rc\n";
                                }
                                else{
                                        $rc = system("$ddlpath${delim}$loadscript");
                                        #$rc = &dodb2file_noconn("$ddlpath$
{delim}$loadscript $sf");
                                        ($rc == 0) || print "ERROR:[$0] load script
$loadscript failed. rc = $rc\n";
                                }
                }

        elsif ( $mode eq "mpp" ){
                &outtime("*** Starting the load");
                # call the appropriate dbgen/split/(sort)/load for mpp
                if ( $loadscript eq "NULL"){
                        $rc = system("perl genloadmpp $qual");
                        ($rc == 0) || print "ERROR:[$0] genloadmpp failed. rc = $rc\n";
                }
                else{
                        system("$ddlpath${delim}$loadscript");
                        #$rc = &dodb2file_noconn("$ddlpath${delim}$loadscript
$sf");
                        #($rc == 0) || print "ERROR:[$0] load script $loadscript failed.
rc = $rc\n";
                }
        }
                else{
                                print "TPCD_MODE not set to one of uni, smp, mln or
```

```
mpp\n";
                        $rc = 1;
            }
            ($rc == 0) && &outtime("*** Load complete");
            return $rc;
}

#-----------------------------------------------------------------------#
# 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);
            #doug
            if ( $qual eq "QUAL" ){
               system("db2 -tvf $ddlpath${delim}chng_ovrhd_qual.sql");
      }
            else{
               system("db2 -tvf $ddlpath${delim}chng_ovrhd.sql");
            }
      &outtime("*** Runstats completed");
}

#-----------------------------------------------------------------------#
# Function: setConfiguration                                            #
#-----------------------------------------------------------------------#
sub setConfiguration{
            my $ret = 0;
&dodb_noconn("db2 update database manager configuration using
max_querydegree $smpdegree",$once);
            &dodb_noconn("db2 update database configuration for $dbname
using dft_degree $smpdegree",$all_ln);
#mujib      &dodb_noconn("db2 update database manager configuration using
max_querydegree $smpdegree",$once);
&dodb2file_noconn("${ddlpath}${delim}$dbmconfig",$once);
            &dodb2file_noconn("${ddlpath}${delim}$configfile",$all_ln);
#mujib      &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_ln);
            print "\n NOTE:  DO NOT RESET THE DATABASE
CONFIGURATION or you will lose logretain\n";
            # force a connection to the database on all nodes to ensure
LOGRETAIN is
            # set in effect.
            # An error message will print to screen if the logretain is set
properly
            # i.e. SQL116N A connection to or activation of database <database
name>
            # cannot  be made.
            # This is expected and the lack of this error message should be seen
as an
            # error in the database build.
            &dodb_conn($dbname,"db2 \"select count(*) from
tpcd.region\"",$all_ln);

            if ( $qual eq "QUAL" ){
                        &outtime("*** Starting the backup");
                        if (( $mode eq "mln" ) || ( $mode eq "mpp")){
                           # must back up catalog node first...assume node 00

                           #$rc=system("db2_all \'}]<<+000< db2 \"backup
database $dbname to $backupdir without prompting\" \' ");
                           $rc=system("db2_all '}]<<+000<\"typeset -i ln=##;
db2 backup database $dbname to /mnt/backup/node\${ln} WITH 16
BUFFERS PARALLELISM 8 without prompting;'");
                           ($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 $backupdir without prompting\' ");
                        $rc=system ("db2_all '}]<<-000<\"typeset -i ln=##; db2
backup database $dbname to /mnt/backup/node\${ln} WITH 16 BUFFERS
PARALLELISM 8 without prompting;'");
                           ($rc == 0 )|| print "ERROR: backup of remaining
nodes failed rc = $rc\n";
                        }
                        else{
                           $rc = &dodb_noconn("db2 backup database $dbname
to $backupdir",$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("db2dart $dbname /CHST /
WHAT DBBP OFF",$all_ln);

                        &outtime("*** Starting the backup");

                        $rc=system ("db2_all '}]<<+000<\"typeset -i ln=##; db2
backup database tpcd to /mnt/backup/node\${ln} WITH 16 BUFFERS
PARALLELISM 8 without prompting;'");
                        ($rc == 0 ) || print "ERROR: backup of catalog node
failed rc = $rc\n";

                        $rc=system ("db2_all '||}]<<-000<\"typeset -i ln=##; db2
backup database tpcd to /mnt/backup/node\${ln} WITH 16 BUFFERS
PARALLELISM 8 without prompting;'");
                        ($rc == 0 )|| print "ERROR: backup of remaining nodes
failed rc = $rc\n";
```

```
                    &outtime("*** Finished the backup");
        }
        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
$dbpath with: \n";
        print "   Mode = $mode \n";
        print "   Tablespace ddl in $ddlpath${delim}$tbspddl \n";
        if ( $nodegroupdef ne "NULL" ){
          print "   Nodegroup ddl in $ddlpath${delim}$nodegroupdef \n";
        }
        if ( $buffpooldef ne "NULL" ){
          print "   Bufferpool ddl in $ddlpath${delim}$buffpooldef \n";
        }
        print "   Table ddl in $ddlpath${delim}$ddl \n";
        print "   Index ddl in              $ddlpath${delim}$indexddl\n";
        if ( $extraindex ne "no" ){
          print " Indices to create after the load $ddlpath${delim}
$extraindex\n";
        }
        if ( $loadscript eq "NULL"){
           if ( $input eq "NULL" ){
              print "   Data generated by DBGEN in $dbgen\n";
           }
           else{
              print "   Data loaded from flat files in $input\n";
           }
        }
        if ( $earlyindex eq "yes" ){
          print "   Indexes created before loading\n";
        }
        else{
          print "   Indexes created after loading\n";
        }
        if ( $addRI ne "NULL" ){
           print "   RI being used from $ddlpath${delim}$addRI\n";
        }
        if ( $astFile ne "NULL" ){
           print "   AST being used from $ddlpath${delim}$astFile\n";
        }
        if ( $loadstats eq "yes" ){
          if ( $earlyindex eq "yes" ){
            print "   Statistics for tables and indexes gathered during load\n";
          }
          else{
            if ( $runstatShort eq "NULL" ){
              print "   Statistics for tables and indexes gathered after load
using $ddlpath${delim}$runstats \n";
            }
            else{
              print "   Statistics for tables and indexes gathered after load
using $ddlpath${delim}$runstats and $ddlpath${delim}$runstatShort\n";
            }
          }
        }
        else{
          if ( $runstatShort eq "NULL" ){
            print "   Statistics for tables and indexes gathered after load using
$ddlpath${delim}$runstats \n";
          }
          else{
            print "   Statistics for tables and indexes gathered after load using
$ddlpath${delim}$runstats and $ddlpath${delim}$runstatShort\n";
          }
        }
        if ( $loadconfigfile ne "NULL" ){
          print "   Database Configuration parameters for LOAD taken from
$ddlpath${delim}$loadconfigfile\n";
        }
        if ( $loadDBMconfig ne "NULL" ){
          print "   Database manager Configuration parameters for LOAD
taken from $ddlpath${delim}$loadDBMconfig\n";
        }
        if ( $configfile ne "NULL" ){
          print "   Database Configuration parameters taken from $ddlpath$
{delim}$configfile\n";
        }
        else{
          print "   Database Configuration paramters taken from $ddlpath$
{delim}dbcfg_for_run${sfReal}GB\n";
          $configfile="dbcfg_for_run${sfReal}GB";
        }
        if ( $dbmconfig ne "NULL" ){
          print "   Database Manager Configuration parameters taken from
$ddlpath${delim}$dbmconfig\n";
        }
        else{
          print "   Database Manager Configuration paramters taken from
$ddlpath${delim}dss.dbmconfig${sfReal}GB\n";
          $configfile="dss.dbmconfig${sfReal}GB";
        }
        #print "   Copy image for load command created in $copydir\n";
        if ( $log eq "yes" ){
          print "   Backup files placed in $backupdir\n";
        }
        else{
          print "   No backup will be taken.\n";
        }
        print "   Log retain set to $log\n";
        if ( $logDir eq "NULL" ){
           print "   Log files remain in database path\n";
        }
        else{
           print "   Log file path set to $logDir\n";
        }
        if ( $logprimary eq "NULL" ){
           print "   Log Primary left at default\n";
        }
        else{
           print "   Log Primary set to $logprimary\n";
        }
        if ( $logsecond eq "NULL" ){
           print "   Log Second left at default\n";
        }
        else{
           print "   Log second set to $logsecond\n";
        }
        if ( $logfilsiz eq "NULL" ){
           print "   Logfilsiz left at default\n";
        }
        else{
           print "   Logfilsiz set to $logfilsiz\n";
        }
        if (($loadconfigfile eq "") || ($loadconfigfile eq "NULL")){
          print "   Machine size set to $machine so the following
configuration\n";
          print "   parameters are used for load, create index and runstats:
\n";
          print "      BUFFPAGE = $buffpage \n";
          print "      SORTHEAP = $sortheap \n";
          print "      SHEAPTHRES = $sheapthres\n";
          print "      NUM_IOSERVERS = $ioservers\n";
          print "      NUM_IOCLEANERS = $ioclnrs\n";
          print "      CHNGPGS_THRESH = $chngpgs\n";
          print "      UTIL_HEAP_SZ = $util_heap_sz\n";
          print "      Degree of parallelism (dft_degree and
max_querydegree) set to $smpdegree\n";
          print "      Parameters for load are: temp file      = $ldtemp\n";
          print "                               sort buf       = $sortbuf\n";
          print "                               ld parallelism = $load_parallelism\n";
          if ( $fparse eq "yes" ){
            print "                               FASTPARSE used on load\n";
          }
        }
        if ( $loadscript ne "NULL"){
          print "   Load commands  in $ddlpath${delim}$loadscript\n";
        }
        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;
```

---

load_db2st

```
#!/bin/ksh
db2set DB2OPTIONS="-t -v +c"
db2set DB2_EXTENDED_OPTIMIZATION=Y
db2set DB2_ANTIJOIN=Y
db2set DB2BPVARS="/mnt/disk2/tpch/tpcd/samples/custom.linux/bpvars.cfg"
db2set DB2_PARALLEL_IO="*"
db2set DB2_STRIPED_CONTAINERS=ON
#db2set DB2NOLIOAIO=false
#db2set DB2NOLIOAIO=true
#db2set DB2LINUXAIO=FALSE
db2set DB2LINUXAIO=TRUE
db2set DB2COMM=tcpip
#db2set DB2_LGPAGE_BP=YES
#db2set DB2_LGPAGE_BP=NO
db2set DB2CODEPAGE=819
```

---

load_dbmcfg.ddl

```
UPDATE DBM CFG USING CPUSPEED -1
DIAGLEVEL 3
SHEAPTHRES 211200
MAX_QUERYDEGREE ANY
INTRA_PARALLEL NO
SVCENAME xtpch
NUMDB 1
DFT_MON_TIMESTAMP OFF
num_poolagents 4
num_initagents 4
agent_stack_sz 16
aslheapsz 15
rqrioblk 32767
maxagents 200;
```

---

load_dbcfg.ddl

```
UPDATE DB CFG FOR TPCD USING DBHEAP 15000
SORTHEAP 25000
SHEAPTHRES_SHR 0
APPGROUP_MEM_SZ 2000
DFT_QUERYOPT 7
DFT_DEGREE 4
NUM_FREQVALUES 0
NUM_QUANTILES 300
LOCKLIST 16384
MAXLOCKS 60
CHNGPGS_THRESH 15
NUM_IOCLEANERS 4
NUM_IOSERVERS 8
MAXFILOP 1024
LOGFILSIZ 8192
LOGPRIMARY 10
LOGSECOND 5
SOFTMAX 600
DATABASE_MEMORY AUTOMATIC
UTIL_HEAP_SZ 50000
buffpage 5000;
```

---

create_nodegroups.ddl

```
create nodegroup ng_all on nodes (0 to 7);
create nodegroup ng_node0 on node (0);
```

---

change_logs.ksh

```
#!/usr/bin/ksh

db2_all "<<+0< db2 update db cfg for tpcd using newlogpath /dev/raw/raw1"
db2_all "<<+1< db2 update db cfg for tpcd using newlogpath /dev/raw/raw2"
db2_all "<<+2< db2 update db cfg for tpcd using newlogpath /dev/raw/raw3"
db2_all "<<+3< db2 update db cfg for tpcd using newlogpath /dev/raw/raw4"
db2_all "<<+4< db2 update db cfg for tpcd using newlogpath /dev/raw/raw5"
db2_all "<<+5< db2 update db cfg for tpcd using newlogpath /dev/raw/raw6"
db2_all "<<+6< db2 update db cfg for tpcd using newlogpath /dev/raw/raw7"
db2_all "<<+7< db2 update db cfg for tpcd using newlogpath /dev/raw/raw8"
```

---

create_bufferpools.ddl

```
-------------------------------------------------------------
-- Create Bufferpools
-------------------------------------------------------------
ALTER BUFFERPOOL IBMDEFAULTBP SIZE 50000;
COMMIT WORK;
CREATE BUFFERPOOL BP32K SIZE 140000 PAGESIZE 32K
numblockpages 80000 blocksize 16;
COMMIT WORK;
create bufferpool bp32ktemp ALL NODES SIZE 20000 PAGESIZE 32K;
COMMIT WORK;
```

---

createuftbl

```
-------------------------------------------------------------
-- 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 other_data;

CREATE TABLE TPCDTEMP.ORDERS_DEL (
APP_ID INTEGER NOT NULL,
O_ORDERKEY INTEGER NOT NULL)
PARTITIONING KEY (O_ORDERKEY)
IN other_data;

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 line_data;

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;
```

---

preloadUF.ksh

```
#!/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=/mnt/disk2/tpch/tpcd/tools

cd $toolsDir
./ploaduf1 32
./ploaduf2 32

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"
```

---

ploaduf1

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

---

ploaduf2

```
#!/bin/ksh
RFpair=$1;
db2 connect to tpcd
db2 "load from delete.new.$RFpair of del modified by coldell fastparse
messages /dev/null replace into TPCDTEMP.ORDERS_DEL nonrecoverable
partitioned db config mode load_only part_file_location /flatfiles_1/ufdata;"
db2 commit;
db2 connect reset
db2 terminate
```

DeleteSampleUFData.sql

```
connect to tpcd;
delete from TPCDTEMP.ORDERS_NEW;
delete from TPCDTEMP.LINEITEM_NEW;
delete from TPCDTEMP.ORDERS_DEL;
commit work;
connect reset;
```

---

create_tablespaces.ddl

```
-----------------------------------------------------------------
-- Create Tablespaces
-----------------------------------------------------------------
CREATE temporary TABLESPACE tempspace
PAGESIZE 32K
MANAGED BY database
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node0_temp32k_1'  1048576,
DEVICE '/mnt/disk2/tpch/tpcd/links/node0_temp32k_2'  1048576
) ON NODE (0)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node1_temp32k_1'  1048576,
DEVICE '/mnt/disk2/tpch/tpcd/links/node1_temp32k_2'  1048576
) ON NODE (1)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node2_temp32k_1' 1048576,
DEVICE '/mnt/disk2/tpch/tpcd/links/node2_temp32k_2' 1048576
) ON NODE (2)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node3_temp32k_1' 1048576,
DEVICE '/mnt/disk2/tpch/tpcd/links/node3_temp32k_2' 1048576
) ON NODE (3)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node4_temp32k_1' 1048576,
DEVICE '/mnt/disk2/tpch/tpcd/links/node4_temp32k_2' 1048576
) ON NODE (4)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node5_temp32k_1' 1048576,
DEVICE '/mnt/disk2/tpch/tpcd/links/node5_temp32k_2' 1048576
) ON NODE (5)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node6_temp32k_1' 1048576,
DEVICE '/mnt/disk2/tpch/tpcd/links/node6_temp32k_2' 1048576
) ON NODE (6)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node7_temp32k_1' 1048576,
```

```
DEVICE '/mnt/disk2/tpch/tpcd/links/node7_temp32k_2' 1048576
) ON NODE (7)
BUFFERPOOL BP32KTEMP
EXTENTSIZE 16
PREFETCHSIZE 256;
COMMIT WORK;

CREATE regular TABLESPACE small_data
IN NODEGROUP ng_node0
PAGESIZE 4K
MANAGED BY database
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/small_data' 262140 )
EXTENTSIZE 16
prefetchsize 256
BUFFERPOOL ibmdefaultbp;
COMMIT WORK;

CREATE regular TABLESPACE line_data
IN NODEGROUP ng_all
PAGESIZE 32K
MANAGED BY database
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node0_line_data_1'  570000,
DEVICE '/mnt/disk2/tpch/tpcd/links/node0_line_data_2'  570000
) ON NODE (0)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node1_line_data_1'  570000,
DEVICE '/mnt/disk2/tpch/tpcd/links/node1_line_data_2'  570000
) ON NODE (1)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node2_line_data_1' 570000,
DEVICE '/mnt/disk2/tpch/tpcd/links/node2_line_data_2' 570000
) ON NODE (2)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node3_line_data_1' 570000,
DEVICE '/mnt/disk2/tpch/tpcd/links/node3_line_data_2' 570000
) ON NODE (3)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node4_line_data_1' 570000,
DEVICE '/mnt/disk2/tpch/tpcd/links/node4_line_data_2' 570000
) ON NODE (4)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node5_line_data_1' 570000,
DEVICE '/mnt/disk2/tpch/tpcd/links/node5_line_data_2' 570000
) ON NODE (5)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node6_line_data_1' 570000,
DEVICE '/mnt/disk2/tpch/tpcd/links/node6_line_data_2' 570000
) ON NODE (6)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node7_line_data_1' 570000,
DEVICE '/mnt/disk2/tpch/tpcd/links/node7_line_data_2' 570000
) ON NODE (7)
BUFFERPOOL BP32K
EXTENTSIZE 16
PREFETCHSIZE 256;
COMMIT WORK;

CREATE temporary TABLESPACE tempspace2
PAGESIZE 4K
MANAGED BY database
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node0_temp4k_1'  4020250,
DEVICE '/mnt/disk2/tpch/tpcd/links/node0_temp4k_2'  4020250
) ON NODE (0)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node1_temp4k_1' 4020250,
DEVICE '/mnt/disk2/tpch/tpcd/links/node1_temp4k_2' 4020250
) ON NODE (1)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node2_temp4k_1' 4020250,
DEVICE '/mnt/disk2/tpch/tpcd/links/node2_temp4k_2' 4020250
) ON NODE (2)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node3_temp4k_1' 4020250,
DEVICE '/mnt/disk2/tpch/tpcd/links/node3_temp4k_2' 4020250
) ON NODE (3)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node4_temp4k_1' 4020250,
DEVICE '/mnt/disk2/tpch/tpcd/links/node4_temp4k_2' 4020250
) ON NODE (4)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node5_temp4k_1' 4020250,
DEVICE '/mnt/disk2/tpch/tpcd/links/node5_temp4k_2' 4020250
) ON NODE (5)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node6_temp4k_1' 4020250,
DEVICE '/mnt/disk2/tpch/tpcd/links/node6_temp4k_2' 4020250
) ON NODE (6)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node7_temp4k_1' 4020250,
DEVICE '/mnt/disk2/tpch/tpcd/links/node7_temp4k_2' 4020250
) ON NODE (7)
BUFFERPOOL ibmdefaultbp
EXTENTSIZE 128
PREFETCHSIZE 256;
COMMIT WORK;

CREATE regular TABLESPACE line_index
IN NODEGROUP ng_all
PAGESIZE 32K
MANAGED BY database
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node0_line_index_1'  70000,
DEVICE '/mnt/disk2/tpch/tpcd/links/node0_line_index_2'  70000
) ON NODE (0)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node1_line_index_1'  70000,
DEVICE '/mnt/disk2/tpch/tpcd/links/node1_line_index_2'  70000
) ON NODE (1)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node2_line_index_1' 70000,
DEVICE '/mnt/disk2/tpch/tpcd/links/node2_line_index_2' 70000
) ON NODE (2)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node3_line_index_1' 70000,
DEVICE '/mnt/disk2/tpch/tpcd/links/node3_line_index_2' 70000
) ON NODE (3)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node4_line_index_1' 70000,
DEVICE '/mnt/disk2/tpch/tpcd/links/node4_line_index_2' 70000
) ON NODE (4)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node5_line_index_1' 70000,
DEVICE '/mnt/disk2/tpch/tpcd/links/node5_line_index_2' 70000
) ON NODE (5)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node6_line_index_1' 70000,
DEVICE '/mnt/disk2/tpch/tpcd/links/node6_line_index_2' 70000
) ON NODE (6)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node7_line_index_1' 70000,
DEVICE '/mnt/disk2/tpch/tpcd/links/node7_line_index_2' 70000
) ON NODE (7)
BUFFERPOOL BP32K
EXTENTSIZE 16
PREFETCHSIZE 256;
COMMIT WORK;

CREATE regular TABLESPACE other_data
IN NODEGROUP ng_all
PAGESIZE 32K
MANAGED BY database
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node0_other_data_1'  260000,
DEVICE '/mnt/disk2/tpch/tpcd/links/node0_other_data_2'  260000
) ON NODE (0)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node1_other_data_1'  260000,
DEVICE '/mnt/disk2/tpch/tpcd/links/node1_other_data_2'  260000
) ON NODE (1)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node2_other_data_1' 260000,
DEVICE '/mnt/disk2/tpch/tpcd/links/node2_other_data_2' 260000
) ON NODE (2)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node3_other_data_1' 260000,
DEVICE '/mnt/disk2/tpch/tpcd/links/node3_other_data_2' 260000
) ON NODE (3)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node4_other_data_1' 260000,
DEVICE '/mnt/disk2/tpch/tpcd/links/node4_other_data_2' 260000
) ON NODE (4)
```

```
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node5_other_data_1' 260000,
DEVICE '/mnt/disk2/tpch/tpcd/links/node5_other_data_2' 260000
) ON NODE (5)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node6_other_data_1' 260000,
DEVICE '/mnt/disk2/tpch/tpcd/links/node6_other_data_2' 260000
) ON NODE (6)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node7_other_data_1' 260000,
DEVICE '/mnt/disk2/tpch/tpcd/links/node7_other_data_2' 260000
) ON NODE (7)
BUFFERPOOL BP32K
EXTENTSIZE 16
PREFETCHSIZE 256;
COMMIT WORK;

CREATE regular TABLESPACE other_index
IN NODEGROUP ng_all
PAGESIZE 32K
MANAGED BY database
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node0_other_index_1'  55000,
DEVICE '/mnt/disk2/tpch/tpcd/links/node0_other_index_2'  55000
) ON NODE (0)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node1_other_index_1'  55000,
DEVICE '/mnt/disk2/tpch/tpcd/links/node1_other_index_2'  55000
) ON NODE (1)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node2_other_index_1' 55000,
DEVICE '/mnt/disk2/tpch/tpcd/links/node2_other_index_2' 55000
) ON NODE (2)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node3_other_index_1' 55000,
DEVICE '/mnt/disk2/tpch/tpcd/links/node3_other_index_2' 55000
) ON NODE (3)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node4_other_index_1' 55000,
DEVICE '/mnt/disk2/tpch/tpcd/links/node4_other_index_2' 55000
) ON NODE (4)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node5_other_index_1' 55000,
DEVICE '/mnt/disk2/tpch/tpcd/links/node5_other_index_2' 55000
) ON NODE (5)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node6_other_index_1' 55000,
DEVICE '/mnt/disk2/tpch/tpcd/links/node6_other_index_2' 55000
) ON NODE (6)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/node7_other_index_1' 55000,
DEVICE '/mnt/disk2/tpch/tpcd/links/node7_other_index_2' 55000
) ON NODE (7)
BUFFERPOOL BP32K
EXTENTSIZE 16
PREFETCHSIZE 256;
COMMIT WORK;


drop tablespace tempspace1;
commit work;


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

create_tables.ddl

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


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

CREATE TABLE TPCD.PART  ( P_PARTKEY     INTEGER NOT NULL,
                    P_NAME        VARCHAR(55) NOT NULL,
                    P_MFGR        CHAR(25) NOT NULL,
                    P_BRAND       CHAR(10) NOT NULL,
                    P_TYPE        VARCHAR(25) NOT NULL,
                    P_SIZE        INTEGER NOT NULL,
                    P_CONTAINER   CHAR(10) NOT NULL,
                    P_RETAILPRICE FLOAT NOT NULL,
                    P_COMMENT     VARCHAR(23) NOT NULL WITH
DEFAULT )
      IN other_data
      INDEX IN other_index
      PARTITIONING KEY(P_PARTKEY) USING HASHING;

CREATE TABLE TPCD.SUPPLIER ( S_SUPPKEY     INTEGER NOT
NULL,
                    S_NAME        CHAR(25) NOT NULL,
                    S_ADDRESS     VARCHAR(40) NOT NULL,
                    S_NATIONKEY   INTEGER NOT NULL,
                    S_PHONE       CHAR(15) NOT NULL,
                    S_ACCTBAL     FLOAT NOT NULL,
                    S_COMMENT     VARCHAR(101) NOT NULL WITH
DEFAULT)
      IN other_data
      INDEX IN other_index
      PARTITIONING KEY(S_SUPPKEY) USING HASHING
      ORGANIZE BY (
       ( "S_NATIONKEY" ) );

CREATE TABLE TPCD.PARTSUPP ( PS_PARTKEY     INTEGER NOT
NULL,
                    PS_SUPPKEY     INTEGER NOT NULL,
                    PS_AVAILQTY    INTEGER NOT NULL,
                    PS_SUPPLYCOST  FLOAT   NOT NULL,
                    PS_COMMENT     VARCHAR(199) NOT NULL WITH
DEFAULT)
      IN other_data
      INDEX IN other_index
      PARTITIONING KEY(PS_PARTKEY) USING HASHING;

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

CREATE TABLE TPCD.ORDERS ( O_ORDERKEY       INTEGER NOT
NULL,
                    O_CUSTKEY        INTEGER NOT NULL,
                    O_ORDERSTATUS    CHAR(1) NOT NULL,
                    O_TOTALPRICE     FLOAT NOT NULL,
                    O_ORDERDATE      DATE NOT NULL,
                    O_ORDERPRIORITY  CHAR(15) NOT NULL,
                    O_CLERK          CHAR(15) NOT NULL,
                    O_SHIPPRIORITY   INTEGER NOT NULL,
                    O_COMMENT        VARCHAR(79) NOT NULL WITH
DEFAULT
--                    PRIMARY KEY (O_ORDERKEY)
                    )
      IN other_data
      INDEX IN other_index
      PARTITIONING KEY(O_ORDERKEY) USING HASHING
      ORGANIZE BY ( ( O_ORDERDATE ) );

CREATE TABLE TPCD.LINEITEM ( L_ORDERKEY    INTEGER NOT
NULL,
                    L_PARTKEY     INTEGER NOT NULL,
                    L_SUPPKEY     INTEGER NOT NULL,
                    L_LINENUMBER  INTEGER NOT NULL,
```

```
                L_QUANTITY    FLOAT NOT NULL,
                L_EXTENDEDPRICE  FLOAT NOT NULL,
                L_DISCOUNT    FLOAT NOT NULL,
                L_TAX         FLOAT NOT NULL,
                L_RETURNFLAG  CHAR(1) NOT NULL,
                L_LINESTATUS  CHAR(1) NOT NULL,
                L_SHIPDATE    DATE NOT NULL,
                L_COMMITDATE  DATE NOT NULL,
                L_RECEIPTDATE DATE NOT NULL,
                L_SHIPINSTRUCT CHAR(25) NOT NULL,
                L_SHIPMODE     CHAR(10) NOT NULL,
                L_COMMENT      VARCHAR(44) NOT NULL WITH
DEFAULT
--                PRIMARY KEY (L_ORDERKEY, L_LINENUMBER)
                )
    IN line_data
    INDEX IN line_index
    PARTITIONING KEY(L_ORDERKEY) USING HASHING
    ORGANIZE BY (  (L_SHIPDATE) );

COMMIT WORK;
```

---

load_database.csh

```
-----------------------------------
#!/bin/csh

set load_script_path = /mnt/disk2/tpch/tpcd/samples/custom.linux/load
set load_log_path = /mnt/disk2/tpch/tpcd/load_log

echo '=========================================='  >>
${load_log_path}/load_database.$$.log
echo 'Starting the load at : `date`            >> ${load_log_path}/
load_database.$$.log
echo '=========================================='  >>
${load_log_path}/load_database.$$.log


db2 -tvf ${load_script_path}/load_all.sql >> ${load_log_path}/
load_database.$$.log


echo '=========================================='  >>
${load_log_path}/load_database.$$.log
echo 'Load complete at : `date`            >> ${load_log_path}/
load_database.$$.log
echo '=========================================='  >>
${load_log_path}/load_database.$$.log

exit
```

---

load_all.sql

```
-----------------------------------
connect to tpcd;

load from
/flatfiles_1/all_node_flatfiles/lineitem.tbl.1,
/flatfiles_1/all_node_flatfiles/lineitem.tbl.2,
/flatfiles_1/all_node_flatfiles/lineitem.tbl.3,
/flatfiles_1/all_node_flatfiles/lineitem.tbl.4,
/flatfiles_1/all_node_flatfiles/lineitem.tbl.5,
/flatfiles_1/all_node_flatfiles/lineitem.tbl.6,
/flatfiles_1/all_node_flatfiles/lineitem.tbl.7,
/flatfiles_1/all_node_flatfiles/lineitem.tbl.8,
/flatfiles_1/all_node_flatfiles/lineitem.tbl.9,
/flatfiles_1/all_node_flatfiles/lineitem.tbl.10,
/flatfiles_1/all_node_flatfiles/lineitem.tbl.11,
/flatfiles_1/all_node_flatfiles/lineitem.tbl.12,
/flatfiles_1/all_node_flatfiles/lineitem.tbl.13,
/flatfiles_1/all_node_flatfiles/lineitem.tbl.14,
/flatfiles_1/all_node_flatfiles/lineitem.tbl.15,
/flatfiles_1/all_node_flatfiles/lineitem.tbl.16
OF DEL MODIFIED BY COLDEL| fastparse MESSAGES /
mnt/disk2/tpch/tpcd/load_log/lineitem.msg
REPLACE INTO TPCD.lineitem  NONRECOVERABLE
CPU_PARALLELISM 8
PARTITIONED DB CONFIG MODE LOAD_ONLY
OUTPUT_DBPARTNUMS (0,1,2,3,4,5,6,7)
part_file_location /flatfiles_1/all_node_flatfiles;

commit work;

load from      /flatfiles_1/all_node_flatfiles/orders.tbl
      OF DEL MODIFIED BY COLDEL| fastparse MESSAGES /
mnt/disk2/tpch/tpcd/load_log/orders.msg
      REPLACE INTO TPCD.orders  NONRECOVERABLE
CPU_PARALLELISM 8
      PARTITIONED DB CONFIG MODE LOAD_ONLY
OUTPUT_DBPARTNUMS (0,1,2,3,4,5,6,7)
      part_file_location /flatfiles_1/all_node_flatfiles;


commit work;

load from      /flatfiles_1/all_node_flatfiles/part.tbl.1 ,
                     /flatfiles_1/all_node_flatfiles/part.tbl.2 ,
                     /flatfiles_1/all_node_flatfiles/part.tbl.3 ,
                     /flatfiles_1/all_node_flatfiles/part.tbl.4 ,
                     /flatfiles_1/all_node_flatfiles/part.tbl.5 ,
                     /flatfiles_1/all_node_flatfiles/part.tbl.6 ,
                     /flatfiles_1/all_node_flatfiles/part.tbl.7 ,
                     /flatfiles_1/all_node_flatfiles/part.tbl.8 ,
                     /flatfiles_1/all_node_flatfiles/part.tbl.9 ,
                     /flatfiles_1/all_node_flatfiles/part.tbl.10 ,
                     /flatfiles_1/all_node_flatfiles/part.tbl.11 ,
                     /flatfiles_1/all_node_flatfiles/part.tbl.12 ,
                     /flatfiles_1/all_node_flatfiles/part.tbl.13 ,
                     /flatfiles_1/all_node_flatfiles/part.tbl.14 ,
                     /flatfiles_1/all_node_flatfiles/part.tbl.15 ,
                     /flatfiles_1/all_node_flatfiles/part.tbl.16
      OF DEL MODIFIED BY COLDEL| fastparse MESSAGES /
mnt/disk2/tpch/tpcd/load_log/part.msg
      REPLACE INTO TPCD.part  NONRECOVERABLE
CPU_PARALLELISM 8
      PARTITIONED DB CONFIG MODE LOAD_ONLY
OUTPUT_DBPARTNUMS (0,1,2,3,4,5,6,7)
      part_file_location /flatfiles_1/all_node_flatfiles;


commit work;

load from      /flatfiles_1/all_node_flatfiles/partsupp.tbl.1 ,
                     /flatfiles_1/all_node_flatfiles/partsupp.tbl.2 ,
                     /flatfiles_1/all_node_flatfiles/partsupp.tbl.3 ,
                     /flatfiles_1/all_node_flatfiles/partsupp.tbl.4 ,
                     /flatfiles_1/all_node_flatfiles/partsupp.tbl.5 ,
                     /flatfiles_1/all_node_flatfiles/partsupp.tbl.6 ,
                     /flatfiles_1/all_node_flatfiles/partsupp.tbl.7 ,
                     /flatfiles_1/all_node_flatfiles/partsupp.tbl.8 ,
                     /flatfiles_1/all_node_flatfiles/partsupp.tbl.9 ,
                     /flatfiles_1/all_node_flatfiles/partsupp.tbl.10 ,
                     /flatfiles_1/all_node_flatfiles/partsupp.tbl.11 ,
                     /flatfiles_1/all_node_flatfiles/partsupp.tbl.12 ,
                     /flatfiles_1/all_node_flatfiles/partsupp.tbl.13 ,
                     /flatfiles_1/all_node_flatfiles/partsupp.tbl.14 ,
                     /flatfiles_1/all_node_flatfiles/partsupp.tbl.15 ,
                     /flatfiles_1/all_node_flatfiles/partsupp.tbl.16
 OF DEL MODIFIED BY COLDEL| fastparse MESSAGES /
mnt/disk2/tpch/tpcd/load_log/partsupp.msg
 REPLACE INTO TPCD.partsupp  NONRECOVERABLE
CPU_PARALLELISM 8
 PARTITIONED DB CONFIG MODE LOAD_ONLY
OUTPUT_DBPARTNUMS (0,1,2,3,4,5,6,7)
 part_file_location /flatfiles_1/all_node_flatfiles;

commit work;

load from      /flatfiles_1/all_node_flatfiles/customer.tbl.1 ,
                     /flatfiles_1/all_node_flatfiles/customer.tbl.2 ,
                     /flatfiles_1/all_node_flatfiles/customer.tbl.3 ,
                     /flatfiles_1/all_node_flatfiles/customer.tbl.4 ,
                     /flatfiles_1/all_node_flatfiles/customer.tbl.5 ,
                     /flatfiles_1/all_node_flatfiles/customer.tbl.6 ,
                     /flatfiles_1/all_node_flatfiles/customer.tbl.7 ,
                     /flatfiles_1/all_node_flatfiles/customer.tbl.8 ,
```

```
                        /flatfiles_1/all_node_flatfiles/customer.tbl.9 ,
                        /flatfiles_1/all_node_flatfiles/customer.tbl.10 ,
                        /flatfiles_1/all_node_flatfiles/customer.tbl.11 ,
                        /flatfiles_1/all_node_flatfiles/customer.tbl.12 ,
                        /flatfiles_1/all_node_flatfiles/customer.tbl.13 ,
                        /flatfiles_1/all_node_flatfiles/customer.tbl.14 ,
                        /flatfiles_1/all_node_flatfiles/customer.tbl.15 ,
                        /flatfiles_1/all_node_flatfiles/customer.tbl.16
 OF DEL MODIFIED BY COLDEL| fastparse MESSAGES /
mnt/disk2/tpch/tpcd/load_log/customer.msg
 REPLACE INTO TPCD.customer  NONRECOVERABLE
CPU_PARALLELISM 8
 PARTITIONED DB CONFIG MODE LOAD_ONLY
OUTPUT_DBPARTNUMS (0,1,2,3,4,5,6,7)
 part_file_location /flatfiles_1/all_node_flatfiles;


commit work;

load from       /flatfiles_1/all_node_flatfiles/supplier.tbl.1 ,
                        /flatfiles_1/all_node_flatfiles/supplier.tbl.2 ,
                        /flatfiles_1/all_node_flatfiles/supplier.tbl.3 ,
                        /flatfiles_1/all_node_flatfiles/supplier.tbl.4 ,
                        /flatfiles_1/all_node_flatfiles/supplier.tbl.5 ,
                        /flatfiles_1/all_node_flatfiles/supplier.tbl.6 ,
                        /flatfiles_1/all_node_flatfiles/supplier.tbl.7 ,
                        /flatfiles_1/all_node_flatfiles/supplier.tbl.8 ,
                        /flatfiles_1/all_node_flatfiles/supplier.tbl.9 ,
                        /flatfiles_1/all_node_flatfiles/supplier.tbl.10 ,
                        /flatfiles_1/all_node_flatfiles/supplier.tbl.11 ,
                        /flatfiles_1/all_node_flatfiles/supplier.tbl.12 ,
                        /flatfiles_1/all_node_flatfiles/supplier.tbl.13 ,
                        /flatfiles_1/all_node_flatfiles/supplier.tbl.14 ,
                        /flatfiles_1/all_node_flatfiles/supplier.tbl.15 ,
                        /flatfiles_1/all_node_flatfiles/supplier.tbl.16
OF DEL MODIFIED BY COLDEL| fastparse MESSAGES /
mnt/disk2/tpch/tpcd/load_log/supplier.msg
REPLACE INTO TPCD.supplier  NONRECOVERABLE
CPU_PARALLELISM 8
PARTITIONED DB CONFIG MODE LOAD_ONLY
OUTPUT_DBPARTNUMS (0,1,2,3,4,5,6,7)
part_file_location /flatfiles_1/all_node_flatfiles;


commit work;

load from /flatfiles_1/all_node_flatfiles/nation.tbl
 OF DEL MODIFIED BY COLDEL| fastparse MESSAGES /
mnt/disk2/tpch/tpcd/load_log/nation.msg
 REPLACE INTO TPCD.nation  NONRECOVERABLE;

commit work;

load from /flatfiles_1/all_node_flatfiles/region.tbl
 OF DEL MODIFIED BY COLDEL| fastparse MESSAGES /
mnt/disk2/tpch/tpcd/load_log/region.msg
 REPLACE INTO TPCD.region  NONRECOVERABLE;

commit work;
connect reset;
terminate;


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

create_indexes.ddl

```
-----------------------------------
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);

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

values(current timestamp);
CREATE       INDEX "TPCD    "."PS_SK" ON "TPCD    "."PARTSUPP"
            ("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_PKSK" ON "TPCD
"."PARTSUPP"
            ("PS_PARTKEY" ASC,
             "PS_SUPPKEY" 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 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 6 ;
commit work;

values(current timestamp);
CREATE       INDEX "TPCD  "."O_CK" ON "TPCD    "."ORDERS"
("O_CUSTKEY" ASC)
PCTFREE 6 ;
commit work;

-- values(current timestamp);
-- CREATE       INDEX "TPCD  "."O_OD" ON "TPCD    "."ORDERS"
--              ("O_ORDERDATE" ASC)
```

```
--              PCTFREE 5 ;
-- commit work;

-- values(current timestamp);
-- CREATE       INDEX "TPCD  "."L_OK" ON "TPCD    "."LINEITEM"
--             ("L_ORDERKEY" ASC)
--              PCTFREE 6 ;
-- commit work;

-- values(current timestamp);

-- CREATE       INDEX "TPCD  "."L_PK" ON "TPCD    "."LINEITEM"
--              ("L_PARTKEY" ASC)
--             PCTFREE 6;
--
-- commit work;
-- values(current timestamp);
--
-- CREATE       INDEX "TPCD  "."L_RD" ON "TPCD    "."LINEITEM"
--              ("L_RECEIPTDATE" ASC)
--              PCTFREE 6 ;
-- commit work;
--
-- values(current timestamp);

values(current timestamp);
create unique index tpcd.l_ok_ln on tpcd.lineitem (L_orderkey, l_linenumber)
pctfree 3;
commit work;
values(current timestamp);

alter table tpcd.orders add primary key(o_orderkey);
alter table tpcd.lineitem add primary key(l_orderkey,l_linenumber);
commit work;
```

---

runstats.ddl

```
RUNSTATS ON TABLE TPCD.NATION    WITH DISTRIBUTION on all
columns
  and columns (
    n_name like statistics,
    n_comment like statistics )
  AND   INDEXES ALL;
commit;
RUNSTATS ON TABLE TPCD.REGION    WITH DISTRIBUTION on all
columns
  and columns (
    r_name like statistics,
    r_comment like statistics )
  AND   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   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   INDEXES ALL;
commit;
RUNSTATS ON TABLE TPCD.PARTSUPP  WITH DISTRIBUTION on all
columns
  and columns (
    ps_comment like statistics)
  AND   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   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   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;
```

---

chng_ovrhd.sql

```
connect to tpcd;
alter tablespace line_index overhead 200;
commit work;
connect reset;
terminate;
```

---

run_db2set.ksh

```
#!/bin/ksh
db2set DB2OPTIONS="-t -v +c"
db2set DB2_EXTENDED_OPTIMIZATION=Y
db2set DB2_ANTIJOIN=Y
db2set DB2BPVARS=/mnt/disk2/tpch/tpcd/samples/custom.linux/bpvars.cfg
db2set DB2_PARALLEL_IO="*"
db2set DB2LINUXAIO=Y
db2set DB2_LARGE_PAGE_MEM=DB
db2set DB2CODEPAGE=819
db2set DB2_SCATTERED_IO=ON
```

---

run_dbmcfg.ddl

```
UPDATE DBM CFG USING HEALTH_MON OFF
HEALTH_MON OFF
SHEAPTHRES 140000
MAX_QUERYDEGREE ANY
INTRA_PARALLEL YES
FCM_NUM_BUFFERS 8192
FCM_NUM_RQB 4096
NUM_POOLAGENTS 64
NUM_INITAGENTS 4
CONN_ELAPSE 10
DFT_MON_TIMESTAMP OFF
CPUSPEED 2e-06
MON_HEAP_SZ 90
JAVA_HEAP_SZ 1024
```

```
MAXAGENTS 200
DIAGLEVEL 3
notifylevel 3
COMM_BANDWIDTH 1.00000e+01
INDEXREC ACCESS;
```

---

run_dbcfg.ddl

```
UPDATE DB CFG FOR TPCD USING
DBHEAP 20000
SORTHEAP 10000
DATABASE_MEMORY automatic
UTIL_HEAP_SZ 10000
DFT_QUERYOPT 7
DFT_DEGREE ANY
NUM_FREQVALUES 0
NUM_QUANTILES 300
LOCKLIST 40000
MAXLOCKS 20
CHNGPGS_THRESH 60
NUM_IOCLEANERS 8
NUM_IOSERVERS 16
MAXFILOP 1024
APPLHEAPSZ 16000
APPGROUP_MEM_SZ 5000
STMTHEAP 20000
STAT_HEAP_SZ 10000
DLCHKTIME 5000
buffpage 5000
CATALOGCACHE_SZ 386
LOGBUFSZ 2048
APP_CTL_HEAP_SZ 2058
SHEAPTHRES_SHR 320000
PCKCACHESZ 640
MAXAPPLS 40
LOGFILSIZ 50000
LOGPRIMARY 4
LOGSECOND 2
MINCOMMIT 1
SOFTMAX 100
DFT_PREFETCH_SZ 16;
```

---

load_dbcfg_qual.ddl

```
UPDATE DB CFG FOR TPCDQUAL USING DBHEAP 15000
SORTHEAP 25000
SHEAPTHRES_SHR 0
APPGROUP_MEM_SZ 2000
DFT_QUERYOPT 7
DFT_DEGREE 4
NUM_FREQVALUES 0
NUM_QUANTILES 300
LOCKLIST 16384
MAXLOCKS 60
CHNGPGS_THRESH 15
NUM_IOCLEANERS 4
NUM_IOSERVERS 8
MAXFILOP 1024
LOGFILSIZ 8192
LOGPRIMARY 10
LOGSECOND 5
SOFTMAX 600
DATABASE_MEMORY AUTOMATIC
UTIL_HEAP_SZ 50000
buffpage 5000;
```

---

change_logs_qual.ksh

```
#!/usr/bin/ksh

#db2_all "<<+0< db2 update db cfg for tpcdqual using newlogpath /
dev/raw/raw9"
#db2_all "<<+1< db2 update db cfg for tpcdqual using newlogpath /
dev/raw/raw10"
#db2_all "<<+2< db2 update db cfg for tpcdqual using newlogpath /
dev/raw/raw11"
#db2_all "<<+3< db2 update db cfg for tpcdqual using newlogpath /
dev/raw/raw12"
#db2_all "<<+4< db2 update db cfg for tpcdqual using newlogpath /
dev/raw/raw13"
#db2_all "<<+5< db2 update db cfg for tpcdqual using newlogpath /
dev/raw/raw14"
#db2_all "<<+6< db2 update db cfg for tpcdqual using newlogpath /
dev/raw/raw15"
#db2_all "<<+7< db2 update db cfg for tpcdqual using newlogpath /
dev/raw/raw16"

db2_all "<<+0< db2 update db cfg for tpcdqual using newlogpath /
dev/raw/raw1"
db2_all "<<+1< db2 update db cfg for tpcdqual using newlogpath /
dev/raw/raw2"
db2_all "<<+2< db2 update db cfg for tpcdqual using newlogpath /
dev/raw/raw3"
db2_all "<<+3< db2 update db cfg for tpcdqual using newlogpath /
dev/raw/raw4"
db2_all "<<+4< db2 update db cfg for tpcdqual using newlogpath /
dev/raw/raw5"
db2_all "<<+5< db2 update db cfg for tpcdqual using newlogpath /
dev/raw/raw6"
db2_all "<<+6< db2 update db cfg for tpcdqual using newlogpath /
dev/raw/raw7"
db2_all "<<+7< db2 update db cfg for tpcdqual using newlogpath /
dev/raw/raw8"
```

---

create_tablespaces_qual.ddl

```
-----------------------------------------------------------------
-- Create Tablespaces
-----------------------------------------------------------------
connect to tpcdqual;

CREATE temporary TABLESPACE tempspace
PAGESIZE 32K
MANAGED BY database
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node0_temp32k_1'  65536
) ON NODE (0)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node1_temp32k_1'  65536
) ON NODE (1)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node2_temp32k_1' 65536
) ON NODE (2)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node3_temp32k_1' 65536
) ON NODE (3)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node4_temp32k_1' 65536
) ON NODE (4)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node5_temp32k_1' 65536
) ON NODE (5)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node6_temp32k_1' 65536
) ON NODE (6)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node7_temp32k_1' 65536
) ON NODE (7)
BUFFERPOOL BP32KTEMP
EXTENTSIZE 16
PREFETCHSIZE 256;
COMMIT WORK;
```

```
CREATE regular TABLESPACE small_data
IN NODEGROUP ng_node0
PAGESIZE 4K
MANAGED BY database
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/small_data' 3000 )
EXTENTSIZE 16
prefetchsize 256
BUFFERPOOL ibmdefaultbp;
COMMIT WORK;

CREATE regular TABLESPACE line_data
IN NODEGROUP ng_all
PAGESIZE 32K
MANAGED BY database
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node0_line_data_1'  50000
) ON NODE (0)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node1_line_data_1'  50000
) ON NODE (1)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node2_line_data_1' 50000
) ON NODE (2)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node3_line_data_1' 50000
) ON NODE (3)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node4_line_data_1' 50000
) ON NODE (4)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node5_line_data_1' 50000
) ON NODE (5)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node6_line_data_1' 50000
) ON NODE (6)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node7_line_data_1' 50000
) ON NODE (7)
BUFFERPOOL BP32K
EXTENTSIZE 16
PREFETCHSIZE 256;
COMMIT WORK;

CREATE temporary TABLESPACE tempspace2
PAGESIZE 4K
MANAGED BY database
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node0_temp4k_1'  52428
) ON NODE (0)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node1_temp4k_1' 52428
) ON NODE (1)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node2_temp4k_1' 52428
) ON NODE (2)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node3_temp4k_1' 52428
) ON NODE (3)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node4_temp4k_1' 52428
) ON NODE (4)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node5_temp4k_1' 52428
) ON NODE (5)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node6_temp4k_1' 52428
) ON NODE (6)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node7_temp4k_1' 52428
) ON NODE (7)
BUFFERPOOL ibmdefaultbp
EXTENTSIZE 128
PREFETCHSIZE 256;
COMMIT WORK;

CREATE regular TABLESPACE line_index
IN NODEGROUP ng_all
PAGESIZE 32K
MANAGED BY database
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node0_line_index_1'  1000
) ON NODE (0)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node1_line_index_1'  1000
) ON NODE (1)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node2_line_index_1' 1000
) ON NODE (2)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node3_line_index_1' 1000
) ON NODE (3)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node4_line_index_1' 1000
) ON NODE (4)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node5_line_index_1' 1000
) ON NODE (5)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node6_line_index_1' 1000
) ON NODE (6)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node7_line_index_1' 1000
) ON NODE (7)
BUFFERPOOL BP32K
EXTENTSIZE 16
PREFETCHSIZE 256;
COMMIT WORK;

CREATE regular TABLESPACE other_data
IN NODEGROUP ng_all
PAGESIZE 32K
MANAGED BY database
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node0_other_data_1'  50000
) ON NODE (0)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node1_other_data_1'  50000
) ON NODE (1)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node2_other_data_1' 50000
) ON NODE (2)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node3_other_data_1' 50000
) ON NODE (3)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node4_other_data_1' 50000
) ON NODE (4)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node5_other_data_1' 50000
) ON NODE (5)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node6_other_data_1' 50000
) ON NODE (6)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node7_other_data_1' 50000
) ON NODE (7)
BUFFERPOOL BP32K
EXTENTSIZE 16
PREFETCHSIZE 256;
COMMIT WORK;

CREATE regular TABLESPACE other_index
IN NODEGROUP ng_all
PAGESIZE 32K
MANAGED BY database
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node0_other_index_1'  1000
) ON NODE (0)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node1_other_index_1'  1000
) ON NODE (1)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node2_other_index_1' 1000
) ON NODE (2)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node3_other_index_1' 1000
) ON NODE (3)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node4_other_index_1' 1000
) ON NODE (4)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node5_other_index_1' 1000
```

```
) ON NODE (5)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node6_other_index_1' 1000
) ON NODE (6)
USING (
DEVICE '/mnt/disk2/tpch/tpcd/links/qual/node7_other_index_1' 1000
) ON NODE (7)
BUFFERPOOL BP32K
EXTENTSIZE 16
PREFETCHSIZE 256;
COMMIT WORK;


drop tablespace tempspace1;
commit work;


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

chng_ovrhd_qual.sql

```
-----------------------------------
connect to tpcdqual;
alter tablespace line_index overhead 200;
commit work;
connect reset;
terminate;


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

run_dbcfg_qual.ddl

```
-----------------------------------
UPDATE DB CFG FOR TPCDQUAL USING
DBHEAP 20000
SORTHEAP 10000
DATABASE_MEMORY automatic
UTIL_HEAP_SZ 10000
DFT_QUERYOPT 7
DFT_DEGREE ANY
NUM_FREQVALUES 0
NUM_QUANTILES 300
LOCKLIST 30000
MAXLOCKS 20
CHNGPGS_THRESH 60
NUM_IOCLEANERS 8
NUM_IOSERVERS 16
MAXFILOP 1024
APPLHEAPSZ 16000
APPGROUP_MEM_SZ 5000
STMTHEAP 20000
STAT_HEAP_SZ 10000
DLCHKTIME 5000
buffpage 5000
CATALOGCACHE_SZ 386
LOGBUFSZ 2048
APP_CTL_HEAP_SZ 2058
SHEAPTHRES_SHR 320000
PCKCACHESZ 640
MAXAPPLS 40
LOGFILSIZ 50000
LOGPRIMARY 14
LOGSECOND 0
MINCOMMIT 1
SOFTMAX 500
DFT_PREFETCH_SZ 16;


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

run_dbmcfg_qual.ddl

```
-----------------------------------
UPDATE DBM CFG USING HEALTH_MON OFF
SHEAPTHRES 140000
MAX_QUERYDEGREE ANY
INTRA_PARALLEL YES
FCM_NUM_BUFFERS 8192
FCM_NUM_RQB 4096
NUM_POOLAGENTS 64
NUM_INITAGENTS 4
CONN_ELAPSE 10
DFT_MON_TIMESTAMP OFF
CPUSPEED 2e-06
MON_HEAP_SZ 90
JAVA_HEAP_SZ 1024
MAXAGENTS 200
DIAGLEVEL 3
notifylevel 3
COMM_BANDWIDTH 1.00000e+01
INDEXREC ACCESS;
```

Appendix C: Qualification Queries

Qualification Query Output

QRYQUAL [1-22]
****

QUERY 1

****
Start timestamp 02/23/05 15:00:23.428774

-- 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      55909065222.828              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 02/23/05 15:00:24.715678
Query Time = 1.3 secs
****

QUERY 2

****
Start timestamp 02/23/05 14:59:46.284520

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

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 VSt3rzk3qG698u6ld8HhOByvrTcSTSvQlDQDag 16-886-766-7945 silent pinto beans should have to snooze carefully along the final reques
9847.570 Supplier#000006345 FRANCE 173827 Manufacturer#2 VSt3rzk3qG698u6ld8HhOByvrTcSTSvQlDQDag 16-886-766-7945 silent pinto beans should have to snooze carefully along the final reques
9836.930 Supplier#000007342 RUSSIA 4841 Manufacturer#4 JOlK7C1,7xrEZSSOw 32-399-414-5385 final accounts haggle. bold accounts are furiously dugouts. furiously silent asymptotes are slyly
9817.100 Supplier#000002352 RUSSIA 124815 Manufacturer#2 4LfoHUZjgjEbAKw TgdKcgOc4D4uCYw 32-551-831-1437 blithely pending packages across the ironic accounts grow slyly after the furiously
9817.100 Supplier#000002352 RUSSIA 152351 Manufacturer#3 4LfoHUZjgjEbAKw TgdKcgOc4D4uCYw 32-551-831-1437 blithely pending packages across the ironic accounts grow slyly after the furiously
9739.860 Supplier#000003384 FRANCE 138357 Manufacturer#2 o,Z3v4POifevE k9U1b 6J1ucX,I 16-494-913-5925 slyly ironic theodolites hag
9721.950 Supplier#000008757 UNITED KINGDOM 156241 Manufacturer#3 Atg6GnM4dT2 33-821-407-2995 ironic, even dolphins above the furiously ironic foxes sleep slyly around the caref
9681.330 Supplier#000008406 RUSSIA 78405 Manufacturer#1 ,qUuXcftUl 32-139-873-8571 furiously even deposits affix thinly special theodolites. furiou
9643.550 Supplier#000005148 ROMANIA 107617 Manufacturer#1 kT4ciVFslx9z4s79p Js825 29-252-617-4850 doggedly even ideas boost furiously against the furiously express
9624.820 Supplier#000001816 FRANCE 34306 Manufacturer#3 e7vab91vLJPWxxZnewmnDBpDmxYHrb 16-392-237-6726 blithely regular accounts cajole furiously. regular
9624.780 Supplier#000009658 ROMANIA 189657 Manufacturer#1 oE9uBgEfSS4opIcepXyAYM,x 29-748-876-2014 regular deposits haggle. furiously express asympto
9612.940 Supplier#000003228 ROMANIA 120715 Manufacturer#2 KDdpNKN3cWu7ZSrbdqp7AfSLxx,qWB 29-325-784-8187 carefully pending accounts serve. furiously close deposits boost slyly. q
9612.940 Supplier#000003228 ROMANIA 198189 Manufacturer#4 KDdpNKN3cWu7ZSrbdqp7AfSLxx,qWB 29-325-784-8187 carefully pending accounts serve. furiously close deposits boost slyly. q
9571.830 Supplier#000004305 ROMANIA 179270 Manufacturer#2 qNHZ7WmCzygwMPRDO9Ps 29-973-481-1831 furiously final deposits
9558.100 Supplier#000003532 UNITED KINGDOM 88515 Manufacturer#4 EOeuiiOn21OVpTlGguufFDFsbN1p0lhpxHp 33-152-301-2164 daring, sly accounts breach about th
9492.790 Supplier#000005975 GERMANY 25974 Manufacturer#5 S6mIiCTx82z7lV 17-992-579-4839 always pending packages boost slyly.
9461.050 Supplier#000002536 UNITED KINGDOM 20033 Manufacturer#1 8mmGbyzaU 7ZS2wJumTibypncu9pNkDc4FYA 33-556-973-5522 even foxes are quickly furiously express requests. packages

[rows deleted to save space]

8386.080 Supplier#000008518 FRANCE 36014 Manufacturer#3 2jqzqqAVe9crMVGP,n9nTsQXulNLTUYoJjEDcqWV 16-618-780-7481 slyly ironic theodolites are slyly. dogged, pendin
8376.520 Supplier#000005306 UNITED KINGDOM 190267 Manufacturer#5 9t8Y8 QqSIsoADPt6NLdk,TP5zyRx41oBUlgoGc9 33-632-514-7931 furiously even instructions integrate during the furiously regular re
8348.740 Supplier#000008851 FRANCE 66344 Manufacturer#4 nWxi7GwEbjhw1 16-796-240-2472 ironic instructions nag slyly against the slyly even theodolites. requests alongside of
8338.580 Supplier#000007269 FRANCE 17268 Manufacturer#4 ZwhJSwABUoiB04,3 16-267-277-4365 ruthlessly regular asymptotes a
8328.460 Supplier#000001744 ROMANIA 69237 Manufacturer#5 oLo3fV64q2,FKHa3p,qHnS7Yzv,ps8 29-330-728-5873 blithely silent excuses are slyly above the furiously even courts
8307.930 Supplier#000003142 GERMANY 18139 Manufacturer#1 dqblvV8dCNAorGlJ 17-595-447-6026 theodolites sleep blithely carefully regular warhorses. slyly regular ins
8231.610 Supplier#000009558 RUSSIA 192000 Manufacturer#2 mcdgen,yT1iJDHDS5fV 32-762-137-5858 slyly regular theodolites sleep fluffily express depos
8152.610 Supplier#000002731 ROMANIA 15227 Manufacturer#4 nluXJCuY1tu 29-805-463-2030 gifts use. slyly silent ideas are carefully beneath the silent instructions. slyly sil
8109.090 Supplier#000009186 FRANCE 99185 Manufacturer#1 wgfosrVPexl9pEXWywaqlBMDYYf 16-668-570-1402 quickly pending requests are blithely along the ironic, final requests; instr
8102.620 Supplier#000003347 UNITED KINGDOM 18344 Manufacturer#5 m CtXS2S16i 33-454-274-8532 packages grow special orbits. regular theodolites about the carefully pe
8046.070 Supplier#000008780 FRANCE 191222 Manufacturer#3 AczzuE0UK9osj ,Lx0Jmh 16-473-215-6395 regular epitaphs integrate slyly.
8042.090 Supplier#000003245 RUSSIA 135705 Manufacturer#4 Dh8Ikg39onrbOL4DyTfGw8a9oKUX3d9Y 32-836-132-8872 carefully regular instructions integrate blithely silent foxes. furiously express instructions haggl
8042.090 Supplier#000003245 RUSSIA 150729 Manufacturer#1 Dh8Ikg39onrbOL4DyTfGw8a9oKUX3d9Y 32-836-132-8872 carefully regular instructions integrate blithely silent foxes. furiously express instructions haggl
7992.400 Supplier#000006108 FRANCE 118574 Manufacturer#1 8tBydnTDwUqfBfFV4l3 16-974-998-8937 regular pinto beans are after
7980.650 Supplier#000001288 FRANCE 13784 Manufacturer#4 zE,7HgVPrCn 16-646-464-8247 unusual pinto beans cajole furiously according t
7950.370 Supplier#000008101 GERMANY 33094 Manufacturer#5 kkYvL6IuvojJgTNG IKkaXQDYgx8ILohj 17-627-663-8014 quickly regular requests are furiously. pending deposits wake
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 Y28ITVeYriT3kIGdV2K8fSZ 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 02/23/05 14:59:46.481254
Query Time =            0.2 secs
***************************
```

QUERY 3

```
***************************
Start timestamp 02/23/05 15:00:15.981820

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

-- 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 02/23/05 15:00:20.601433
Query Time =            4.6 secs
***************************
```

QUERY 4

```
***************************
Start timestamp 02/23/05 15:00:22.429512

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

-- 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 02/23/05 15:00:22.872404
Query Time =            0.4 secs
***************************
```

QUERY 5

```
***************************
Start timestamp 02/23/05 15:00:25.782233

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

-- 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 02/23/05 15:00:26.934179
Query Time = 1.2 secs
****

QUERY 6

****
Start timestamp 02/23/05 15:00:06.336400

-- 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 02/23/05 15:00:06.412287
Query Time = 0.1 secs
****

QUERY 7

****
Start timestamp 02/23/05 15:00:26.934179

-- 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 02/23/05 15:00:27.775979
Query Time = 0.8 secs
****

QUERY 8

****
Start timestamp 02/23/05 15:00:08.443561

-- 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 02/23/05 15:00:12.080754
Query Time =          3.6 secs
***************************
```

QUERY 9

```
***************************
Start timestamp 02/23/05 14:59:46.481254

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

-- 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       51923746.575
ARGENTINA                 1995       49298625.767
ARGENTINA                 1994       50835610.109
ARGENTINA                 1993       51646079.177
ARGENTINA                 1992       50410314.995
BRAZIL                 1998       27217924.383
BRAZIL                 1997       48378669.199
BRAZIL                 1996       50482870.357
BRAZIL                 1995       47623383.635
BRAZIL                 1994       47840165.726
BRAZIL                 1993       49054694.035
BRAZIL                 1992       48667639.084
CANADA                 1998       30379833.769
CANADA                 1997       50465052.311
CANADA                 1996       52560501.390
CANADA                 1995       52375332.809
CANADA                 1994       52600364.659

[rows deleted to save space]

SAUDI ARABIA                 1996       52543737.820
SAUDI ARABIA                 1995       52938696.533
SAUDI ARABIA                 1994       51389601.967
SAUDI ARABIA                 1993       52937508.882
SAUDI ARABIA                 1992       54843459.641
UNITED KINGDOM                 1998       28494874.004
UNITED KINGDOM                 1997       49381810.899
UNITED KINGDOM                 1996       51386853.960
UNITED KINGDOM                 1995       51509586.788
UNITED KINGDOM                 1994       48086499.711
UNITED KINGDOM                 1993       49166827.224
UNITED KINGDOM                 1992       49349122.082
UNITED STATES                 1998       25126238.946
UNITED STATES                 1997       50077306.419
UNITED STATES                 1996       48048649.470
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       49658284.612
VIETNAM                 1994       50596057.261
VIETNAM                 1993       50953919.152
VIETNAM                 1992       49613838.315


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


Stop timestamp 02/23/05 15:00:05.773090
Query Time =          19.3 secs
***************************
```

QUERY 10

```
***************************
Start timestamp 02/23/05 15:00:24.715678

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

-- 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         2557.470
EGYPT                1aReFYv,Kw4                         14-742-935-3718
fluffily bold excuses haggle finally after the u
     60838  Customer#0000060838                679127.308         2454.770
BRAZIL               64EaJ5vMAHWJlBOxJklpNc2RJiWE        12-913-
494-9813  furiously even pinto beans integrate under the ruthless foxes; ironic,
even dolphins across the slyl
    101998  Customer#0000101998                637029.567         3790.890
UNITED KINGDOM       01c9CILnNtfOQYmZj                   33-593-
865-6378  accounts doze blithely! enticing, final deposits sleep blithely special
accounts. slyly express accounts pla
    125341  Customer#0000125341                633508.086         4983.510
GERMANY              S29ODD6bceU8QSuuEJznkNaK            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         5098.100
FRANCE               rFeBbEEyk dl ne7zV5fDrmiq1oK09wV7pxqCgIc  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         5583.020
IRAN                 vXCxoCsU0Bad5JQI ,oobkZ             20-834-292-
4707  express requests sublate blithely regular requests. regular, even ideas
solve.
     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,NPGkfyQQ0hcIYUGPIBWc,ybP5g,
33-657-701-3391  quickly final requests against the regular instructions wake
blithely final instructions. pa
       922  Customer#0000000922                576767.533         3869.250
GERMANY              Az9RFaut7NkPnc5zSD2PwHgVwr4jRzq     17-
945-916-9648  boldly final requests cajole blith
    147946  Customer#0000147946                576455.132         2030.130
ALGERIA              iANyZHjqhyy7Ajah0pTrYyhJ            10-886-956-
3143  furiously even accounts are blithely above the furiousl
    115640  Customer#0000115640                569341.193         6436.100
ARGENTINA            Vtgfia9qI 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              HgiV0phqhaIa9aydNoIlb               29-915-458-
2654  instructions nag quickly. furiously bold accounts cajol


Number of rows retrieved is:     20
```

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

Stop timestamp 02/23/05 15:00:25.343886
Query Time =          0.6 secs
**************************
```

QUERY 11

```
**************************
Start timestamp 02/23/05 15:00:22.872404

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

-- 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
     85606        15053957.150
     33354        14408297.400
    154747        14407580.680
     82865        14235489.780
     76094        14094247.040
       222       13937777.740
    121271        13908336.000
     55221        13716120.470
     22819        13666434.280
     76281        13646853.680
     85298        13581154.930
     85158        13554904.000
    139684        13535538.720
     31034        13498025.250
     87305        13482847.040
     10181        13445148.750
     62323        13411824.300

[rows deleted to save space]

     92814         7908262.500
     88919         7907992.500
     79753         7907933.880
    108765         7905338.980
```

```
146530          7905336.600
71475           7903367.580
36289           7901946.500
61739           7900794.000
52338           7898638.080
194299          7898421.240
105235          7897829.940
77207           7897752.720
96712           7897575.270
10157           7897046.250
171154          7896814.500
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 02/23/05 15:00:23.162891
Query Time =            0.3 secs
***************************
```

QUERY 12

```
***************************
Start timestamp 02/23/05 15:00:27.775979

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

-- 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 02/23/05 15:00:28.186787
Query Time =            0.4 secs
***************************
QUERY 13
***************************
Start timestamp 02/23/05 15:00:14.365156

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

-- 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 02/23/05 15:00:15.981820
Query Time =            1.6 secs
***************************
```

QUERY 14

```
***************************
Start timestamp 02/23/05 14:59:45.877740

-------------------------------------------
--#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 02/23/05 14:59:46.284520
Query Time =            0.4 secs
***************************
```

QUERY 15

```
***************************
Start timestamp 02/23/05 15:00:23.162891

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

-- 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 02/23/05 15:00:23.428774
Query Time =            0.3 secs
***************************
```

QUERY 16

```
***************************
Start timestamp 02/23/05 15:00:21.069891

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

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

Tag: Q16    Stream: -1   Sequence number: 13

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

P_BRAND     P_TYPE                    P_SIZE       SUPPLIER_CNT
-----------------------------------------------------------------
Brand#41    MEDIUM BRUSHED TIN              3           28
Brand#54    STANDARD BRUSHED COPPER         14          27
Brand#11    STANDARD BRUSHED TIN            23          24
Brand#11    STANDARD BURNISHED BRASS        36          24
Brand#15    MEDIUM ANODIZED NICKEL          3           24
Brand#15    SMALL ANODIZED BRASS            45          24
Brand#15    SMALL BURNISHED NICKEL          19          24
Brand#21    MEDIUM ANODIZED COPPER          3           24
Brand#22    SMALL BRUSHED NICKEL            3           24
Brand#22    SMALL BURNISHED BRASS           19          24
Brand#25    MEDIUM BURNISHED COPPER         36          24
Brand#31    PROMO POLISHED COPPER           36          24
```

```
Brand#33    LARGE POLISHED TIN              23          24
Brand#33    PROMO POLISHED STEEL            14          24
Brand#35    PROMO BRUSHED NICKEL            14          24
Brand#41    ECONOMY BRUSHED STEEL           9           24
Brand#41    ECONOMY POLISHED TIN            19          24
Brand#41    LARGE PLATED COPPER             36          24
Brand#42    ECONOMY PLATED BRASS            3           24
Brand#42    STANDARD POLISHED TIN           49          24
Brand#43    PROMO BRUSHED TIN               3           24
Brand#43    SMALL ANODIZED COPPER           36          24
Brand#44    STANDARD POLISHED NICKEL        3           24
Brand#52    ECONOMY PLATED TIN              14          24
Brand#52    STANDARD BURNISHED NICKEL       3           24
Brand#53    MEDIUM ANODIZED STEEL           14          24

[rows deleted to save space]

Brand#55    STANDARD POLISHED COPPER        45          4
Brand#55    STANDARD POLISHED NICKEL        3           4
Brand#55    STANDARD POLISHED NICKEL        23          4
Brand#55    STANDARD POLISHED NICKEL        36          4
Brand#55    STANDARD POLISHED NICKEL        45          4
Brand#55    STANDARD POLISHED NICKEL        49          4
Brand#55    STANDARD POLISHED STEEL         14          4
Brand#55    STANDARD POLISHED STEEL         23          4
Brand#55    STANDARD POLISHED TIN           9           4
Brand#55    STANDARD POLISHED TIN           19          4
Brand#55    STANDARD POLISHED TIN           36          4
Brand#11    SMALL BRUSHED TIN               19          3
Brand#15    LARGE PLATED NICKEL             45          3
Brand#15    LARGE POLISHED NICKEL           9           3
Brand#21    PROMO BURNISHED STEEL           45          3
Brand#22    STANDARD PLATED STEEL           23          3
Brand#25    LARGE PLATED STEEL              19          3
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 02/23/05 15:00:22.429512
Query Time =            1.4 secs
***************************
```

QUERY 17

```
***************************
Start timestamp 02/23/05 15:00:06.412287

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

-- 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 02/23/05 15:00:06.801019
Query Time =            0.4 secs
***************************
```

QUERY 18

```
***************************
Start timestamp 02/23/05 15:00:06.801019

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

-- 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
Customer#0000015272           15272     3883783  1993-07-28
500241.330           302.000
Customer#0000146608          146608     3342468  1994-06-12
499794.580           303.000
Customer#0000096103           96103     5984582  1992-03-16
494398.790           312.000
Customer#0000024341           24341     1474818  1992-11-15
491348.260           302.000
```

```
Customer#0000137446          137446     5489475  1997-05-23
487763.250          311.000
Customer#0000107590          107590     4267751  1994-11-04
485141.380          301.000
Customer#0000050008           50008     2366755  1996-12-09
483891.260          302.000
Customer#0000015619           15619     3767271  1996-08-07
480083.960          318.000
Customer#0000077260           77260     1436544  1992-09-12
479499.430          307.000
Customer#0000109379          109379     5746311  1996-10-10
478064.110          302.000
Customer#0000054602           54602     5832321  1997-02-09
471220.080          307.000
Customer#0000105995          105995     2096705  1994-07-03
469692.580          307.000
Customer#0000148885          148885     2942469  1992-05-31
469630.440          313.000
Customer#0000114586          114586      551136  1993-05-19
469605.590          308.000
Customer#0000105260          105260     5296167  1996-09-06
469360.570          303.000
Customer#0000147197          147197     1263015  1997-02-02
467149.670          320.000
Customer#0000064483           64483     2745894  1996-07-04
466991.350          304.000
Customer#0000136573          136573     2761378  1996-05-31
461282.730          301.000
Customer#0000016384           16384      502886  1994-04-12
458378.920          312.000
Customer#0000117919          117919     2869152  1996-06-20
456815.920          317.000
Customer#0000012251           12251      735366  1993-11-24
455107.260          309.000
Customer#0000120098          120098     1971680  1995-06-14
453451.230          308.000
Customer#0000066098           66098     5007490  1992-08-07
453436.160          304.000
Customer#0000117076          117076     4290656  1997-02-05
449545.850          301.000
Customer#0000129379          129379     4720454  1997-06-07
448665.790          303.000
Customer#0000126865          126865     4702759  1994-11-07
447606.650          320.000
Customer#0000088876           88876      983201  1993-12-30
446717.460          304.000
Customer#0000036619           36619     4806726  1995-01-17
446704.090          328.000
Customer#0000141823          141823     2806245  1996-12-29
446269.120          310.000
Customer#0000053029           53029     2662214  1993-08-13
446144.490          302.000
Customer#0000018188           18188     3037414  1995-01-25
443807.220          308.000
Customer#0000066533           66533       29158  1995-10-21
443576.500          305.000
Customer#0000037729           37729     4134341  1995-06-29
441082.970          309.000
Customer#0000003566            3566     2329187  1998-01-04
439803.360          304.000
Customer#0000045538           45538     4527553  1994-05-22
436275.310          305.000
Customer#0000081581           81581     4739650  1995-11-04
435405.900          305.000
Customer#0000119989          119989     1544643  1997-09-20
434568.250          320.000
Customer#0000003680            3680     3861123  1998-07-03
433525.970          301.000
Customer#0000113131          113131      967334  1995-12-15
432957.750          301.000
Customer#0000141098          141098      565574  1995-09-24
430986.690          301.000
Customer#0000093392           93392     5200102  1997-01-22
425487.510          304.000
Customer#0000015631           15631     1845057  1994-05-12
419879.590          302.000
Customer#0000112987          112987     4439686  1996-09-17
418161.490          305.000
Customer#0000012599           12599     4259524  1998-02-12
415200.610          304.000
Customer#0000105410          105410     4478371  1996-03-05
412754.510          302.000
Customer#0000149842          149842     5156581  1994-05-30
411329.350          302.000
Customer#0000010129           10129     5849444  1994-03-21
409129.850          309.000
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 02/23/05 15:00:08.443561
Query Time =          1.6 secs
***************************
```

QUERY 19

```
***************************
Start timestamp 02/23/05 15:00:25.343886

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

-- 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 02/23/05 15:00:25.782233
```

```
Query Time =            0.4 secs
***************************
```

QUERY 20

```
***************************
Start timestamp 02/23/05 15:00:05.773090

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

-- 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
Supplier#000000378        FfbhyCxWvcPrO8ltp9
Supplier#000000402        i9Sw4DoyMhzhKXCH9By,AYSgmD
Supplier#000000530        0qwCMwobKY OcmLyfRXlagA8ukENJv,
Supplier#000000688        D fw5ocppmZpYBBIPI718hCihLDZ5KhKX
Supplier#000000710        f19YPvOyb QoYwjKC,oPycpGfieBAcwKJo
Supplier#000000736        l6i2nMwVuovfKnuVgaSGK2rDy65DlAFLegiL7
Supplier#000000761        zlSLelQUj2XrvTTFnv7WAcYZGvvMTx882d4
Supplier#000000884        bmhEShejaS
Supplier#000000887        urEaTejH5POADP2ARrf
Supplier#000000935        ij98czM 2KzWe7dDTOxB8sq0UfCdvrX
Supplier#000000975        ,AC e,tBpNwKb5xMUzeohxIRn, hdZJo73gFQF8y
Supplier#000001263        rQWr6nf8ZhB2TAiIDIvo5Io
Supplier#000001399        LmrocnIMSyYOWuANx7
Supplier#000001446        lch9HMNU1R7a0LIybsUodVknk6
Supplier#000001454        TOpimgu2TVXIjhiL93h,
Supplier#000001500        wDmF5xLxtQch9ctVu,
Supplier#000001602        uKNWIeafaM644
Supplier#000001626        UhxNRzUu1dtFmp0
Supplier#000001682        pXTkGxrTQVyH1Rr
Supplier#000001699        Q9C4rfJ26oijVPqqcqVXeRI
Supplier#000001700        7hMlCof1Y5zLFg
[rows deleted to save space]

Supplier#000008610        SgVgP90vP452sUNTgzL9zKwXHXAzV6tV
Supplier#000008705        aE,trRNdPx,4yinTD9O3DebDIp
Supplier#000008742        HmPlQEzKCPEcTUL14,kKq
Supplier#000008841        I 85Lu1sekbg2xrSIzm0
Supplier#000008895        2cH4okfaLSZTTg8sKRbbJQxkmeFu2Esj
Supplier#000008967        2kwEHyMG 7FwozNImAUE6mH0hYtqYculJM
Supplier#000008972        w2vF6 D5YZO3visPXsqVfLADTK
Supplier#000009032        qK,trB6Sdy4Dz1BRUFNy
Supplier#000009147        rOAuryHxpZ9eOvx
Supplier#000009252        F7cZaPUHwh1 ZKyj3xmAVWC1XdP ue1p5m,i
Supplier#000009278        RqYTzgxj93CLX 0mcYfCENOefD
Supplier#000009327        uoqMdf7e7Gj9dbQ53
Supplier#000009430        igRqmneFt
Supplier#000009567        r4Wfx4c3xsEAjcGj71HHZByornl D9vrztXlv4
Supplier#000009601        51m637bO,Rw5DnHWFUvLacRx9
Supplier#000009709        rRnCbHYgDgl9PZYnyWKVYSUW0vKg
Supplier#000009753        wLhVEcRmd7PkJF4FBnGK7Z
Supplier#000009796        z,y4Idmr15DOvPUqYG
Supplier#000009799        4wNjXGa4OKWl
Supplier#000009811        E3iuyq7UnZxU7oPZIe2Gu6
Supplier#000009812        APFRMy3lCbgFga53n5t9DxzFPQPgnjrGt32
Supplier#000009862        rJzweWeN58
Supplier#000009868        ROjGgx5gvtkmnUUoeyy7v
Supplier#000009869
ucLqxzrpBTRMewGSM29t0rNTM30g1Tu3Xgg3mKag
Supplier#000009899        7XdpAHrzr1t,UQFZE
Supplier#000009974        7wJ,J5DKcxSU4Kp1cQLpbcAvB5AsvKT


Number of rows retrieved is:    204
-------------------------------------------


Stop timestamp 02/23/05 15:00:06.336400
Query Time =            0.6 secs
***************************
```

QUERY 21

```
***************************
Start timestamp 02/23/05 15:00:12.080754

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

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

Tag: Q21    Stream: -1   Sequence number: 9

select
s_name,
count(*) as numwait
from
tpcd.supplier,
tpcd.lineitem l1,
tpcd.orders,
tpcd.nation
where
s_suppkey = l1.l_suppkey
and o_orderkey = l1.l_orderkey
and o_orderstatus = 'F'
and l1.l_receiptdate > l1.l_commitdate
and exists (
select
*
from
tpcd.lineitem l2
where
l2.l_orderkey = l1.l_orderkey
and l2.l_suppkey <> l1.l_suppkey
)
and not exists (
select
*
from
tpcd.lineitem l3
where
l3.l_orderkey = l1.l_orderkey
and l3.l_suppkey <> l1.l_suppkey
and l3.l_receiptdate > l3.l_commitdate
```

```
)
and s_nationkey = n_nationkey
and n_name = 'SAUDI ARABIA'
group by
s_name
order by
numwait desc,
s_name
fetch first 100 rows only

S_NAME                    NUMWAIT
----------------------------------------
Supplier#000002829            20
Supplier#000005808            18
Supplier#000000262            17
Supplier#000000496            17
Supplier#000002160            17
Supplier#000002301            17
Supplier#000002540            17
Supplier#000003063            17
Supplier#000005178            17
Supplier#000008331            17
Supplier#000002005            16
Supplier#000002095            16
Supplier#000005799            16
Supplier#000005842            16
Supplier#000006450            16
Supplier#000006939            16
Supplier#000009200            16
Supplier#000009727            16
Supplier#000000486            15
Supplier#000000565            15
Supplier#000001046            15
Supplier#000001047            15
Supplier#000001161            15
Supplier#000001336            15
Supplier#000001435            15
Supplier#000003075            15

[rows deleted to save space]

Supplier#000006624            13
Supplier#000006629            13
Supplier#000006682            13
Supplier#000006737            13
Supplier#000006825            13
Supplier#000007021            13
Supplier#000007417            13
Supplier#000007497            13
Supplier#000007602            13
Supplier#000008134            13
Supplier#000008234            13
Supplier#000009435            13
Supplier#000009436            13
Supplier#000009564            13
Supplier#000009896            13
Supplier#000000379            12
Supplier#000000673            12
Supplier#000000762            12
Supplier#000000811            12
Supplier#000000821            12
Supplier#000001337            12
Supplier#000001916            12
Supplier#000001925            12
Supplier#000002039            12
Supplier#000002357            12
Supplier#000002483            12


Number of rows retrieved is:    100
---------------------------------------------


Stop timestamp 02/23/05 15:00:14.365156
Query Time =            2.3 secs
***************************
```

QUERY 22

```
***************************
Start timestamp 02/23/05 15:00:20.601433
---------------------------------------------

-- 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 02/23/05 15:00:21.069891
Query Time =            0.5 secs
```

First 10 Rows

```
  Database Connection Information

 Database server        = DB2/LINUXIA64 8.2.1
 SQL authorization ID   = TPCH
 Local database alias   = TPCD


SELECT * FROM TPCD.REGION FETCH FIRST 10 ROWS ONLY

R_REGIONKEY R_NAME                 R_COMMENT
----------- -------------------------
-----------------------------------------------------------------------------------------
```

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

  5 record(s) selected.


SELECT * FROM TPCD.NATION FETCH FIRST 10 ROWS ONLY

N_NATIONKEY N_NAME                  N_REGIONKEY N_COMMENT
----------- ------------------------- -----------
-----------------------------------------------------------------------------------------------
-----------------------------------------------------
         0 ALGERIA                         0 final accounts wake quickly. special
reques
         5 ETHIOPIA                        0 fluffily ruthless requests integrate
fluffily. pending ideas wake blithely acco
        14 KENYA                           0 ironic requests boost. quickly pending
pinto beans cajole slyly slyly even deposits. ironic packages
        15 MOROCCO                         0 ideas according to the fluffily final
pinto beans sleep furiously
        16 MOZAMBIQUE                      0 ironic courts wake fluffily even,
bold deposi
         1 ARGENTINA                       1 idly final instructions cajole
stealthily. regular instructions wake carefully blithely express accounts. fluffi
         2 BRAZIL                          1 always pending pinto beans sleep sil
         3 CANADA                          1 foxes among the bold requests
        17 PERU                            1 final, final accounts sleep slyly across the
requests.
        24 UNITED STATES                   1 blithely regular deposits serve
furiously blithely regular warthogs! slyly fi

  10 record(s) selected.


SELECT * FROM TPCD.PART FETCH FIRST 10 ROWS ONLY

P_PARTKEY   P_NAME                                         P_MFGR
P_BRAND    P_TYPE                  P_SIZE      P_CONTAINER
P_RETAILPRICE           P_COMMENT
----------- ---------------------------------------------------------------------------------
---------- ------------------------- ----------- ---------- -----------------------
----------------------
         11 chocolate turquoise sandy snow misty           Manufacturer#2
Brand#25   STANDARD BURNISHED NICKEL          43 WRAP BOX
+9.11010000000000E+002 furiousl
         37 turquoise ivory orange sandy maroon            Manufacturer#4
Brand#45   LARGE POLISHED TIN                 48 JUMBO BOX
+9.37030000000000E+002 blithely regular
         43 medium khaki chocolate rosy blush              Manufacturer#4
Brand#44   PROMO POLISHED STEEL               5 WRAP CASE
+9.43040000000000E+002 carefully iro
         47 sky firebrick red linen dim                    Manufacturer#4
Brand#45   LARGE BURNISHED BRASS              14 JUMBO PACK
+9.47040000000000E+002 bold, unusual a
         50 yellow cornflower royal blush almond           Manufacturer#3
Brand#33   LARGE ANODIZED TIN                 25 WRAP PKG
+9.50050000000000E+002 regular dinos ar
         55 antique cream pale tomato rose                 Manufacturer#2
Brand#23   ECONOMY BRUSHED COPPER             9 MED BAG
+9.55050000000000E+002 furiously
         68 bisque frosted pale puff sandy                 Manufacturer#1
Brand#11   PROMO ANODIZED STEEL               10 WRAP BOX
+9.68060000000000E+002 carefully
         76 bisque light yellow puff salmon                Manufacturer#3
Brand#34   MEDIUM BRUSHED COPPER              9 SM PKG
+9.76070000000000E+002 final
         87 pale khaki sandy antique black                 Manufacturer#4
Brand#41   LARGE PLATED STEEL                 41 WRAP PACK
+9.87080000000000E+002 slyly even instruction
         89 ghost khaki lawn pale dim                      Manufacturer#5
Brand#53   STANDARD BURNISHED STEEL           7 MED JAR
+9.89080000000000E+002 quickly ironi

  10 record(s) selected.


SELECT * FROM TPCD.SUPPLIER FETCH FIRST 10 ROWS ONLY

S_SUPPKEY   S_NAME                  S_ADDRESS
S_NATIONKEY S_PHONE         S_ACCTBAL               S_COMMENT
----------- ------------------------- ---------------------------------------- -----------
--------------- -------------------------
-----------------------------------------------------------------------------------------------
----
      1482 Supplier#000001482
vxO8jtZwiIAeNhpIdltdl34JhtC8FndxMXUPj            23 33-579-158-3822
-7.46350000000000E+002 unusual foxes cajole bold excuses. ironic requests
sleep
      1751 Supplier#000001751        VHoWbPbUjo Rdege
23 33-309-349-1317   +4.33254000000000E+003 pending packages haggle
carefully regular instructions. regular account
      2552 Supplier#000002552        ESBwImdv,dqHGsrtrtm9aOm
23 33-183-384-8624   +4.06000000000000E+000 final instructions into the
silent foxes boost accounts. furiously ironic excuses use care
      2874 Supplier#000002874        gAd19gvMltXu6S0DzD76eIAGZ
23 33-345-155-4868   +4.32270000000000E+003 special platelets wake
blithely across the finally silent deposi
      2969 Supplier#000002969        MbZUmRv4UhV59cyNbMCyO
23 33-548-309-7067   +8.15198000000000E+003 carefully final dugouts
affix? slyly even accounts are asymptotes. unusual, special wartho
     16368 Supplier#000016368
QLmIj2qOPG8tVCS,hTCiUwYHBsGbeEDq                 23 33-204-318-9989
+3.64323000000000E+003 bold deposits nag. regular requests according to
the slyly even asymptotes cajole after the regular
     16401 Supplier#000016401
mMHKeFBimg81BKdoWODdBSIsoChAht8oRjP              23 33-274-589-
7111   +2.77947000000000E+003 slyly regular requests use slyly even
packages. theo
     16441 Supplier#000016441        nQobijgABHpcu18H98wVQ
23 33-640-210-7836   +6.31120000000000E+002 carefully unusual foxes are
carefully across
     16513 Supplier#000016513        bwZZWMzPfnz8tOW1qMC
23 33-423-325-5530   +4.91440000000000E+002 slyly ironic instructions nag
quickly
     16795 Supplier#000016795        EZy3C7hMmk1K4 C54qnNo
23 33-275-713-1719   +5.44733000000000E+003 carefully final platelets
solve furiously

  10 record(s) selected.


SELECT * FROM TPCD.PARTSUPP FETCH FIRST 10 ROWS ONLY

PS_PARTKEY  PS_SUPPKEY  PS_AVAILQTY PS_SUPPLYCOST
PS_COMMENT
----------- ----------- ----------- -----------------------
-----------------------------------------------------------------------------------------------
-----------------------------------------------------------------------------------------------
-----
         11          12        4540   +7.09870000000000E+002 final packages mold
after the carefully unusual requests. quickly fi
         11      750012        4729   +8.94900000000000E+002 regular packages
sleep carefully fluffily ironic ac
         11     1500012        3708   +8.18740000000000E+002 slyly pending
theodolites wake quickly unusual, express accounts. fluffily regular requests
cajole furiously quickly even dugouts. slyly bold platelets
         11     2250012        3213   +4.71980000000000E+002 ideas nag regular
instructions. regular, thin pinto beans unwind furiously ironic accounts.
quickly express platele
         37          38        7171   +8.24960000000000E+002 pending packages
wake quietly. special, even accounts haggle unusual
         37      750038        5542   +1.26590000000000E+002 ironic, ironic
packages are slyly. furiously regular excuses cajole blithely quickly ironic
dependencies. regular packages use carefully furiously pending depths
         37     1500038        7113   +1.57200000000000E+001 regular, final
requests nag across the regular, even requests. carefully regular accounts cajole
carefully carefully ruthless dep
         37     2250038        1449   +7.45640000000000E+002 regular, silent
foxes nag. carefully ironic deposits are fluffily. carefully regular pinto beans
around the bravely even instructions cajole carefully furiously regul
         43          44        3211   +8.05780000000000E+002 final, express
dependencies sleep according to the express requests. bold, regular accounts
detect outside the slyly
         43      750044        6770   +4.93190000000000E+002 furiously special
pinto beans cajole. ironic decoys across the
```

```
  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
----------- ------------------------ ---------------------------------------- -----------
--------------- ------------------------ ------------
--------------------------------------------------------------------------------------------
--------------------
         11 Customer#000000011        cG48rYjF3Aw7xs hKUXXqmI
23 33-464-151-3439   -2.72600000000000E+002 BUILDING     furiously
express packages are. regular courts play deposits. silent, ironic packages
engage. furiously regular a
         87 Customer#000000087        KKkJj684aFsUH1Unw
23 33-869-884-7053   +6.32754000000000E+003 FURNITURE    bold,
unusual excuses haggle daringly according to the carefully express requests!
theod
        338 Customer#000000338
sQI2qN4Ki,DGk8hxTcKwhAewbgJulXQF7SHDz3            23 33-302-620-
7535   +4.09249000000000E+003 FURNITURE    carefully final excuses are
sometimes regular instructions.
        364 Customer#000000364        9aXPVbVcy
FE1HhTTyA0DZ7I3h5ha,fT1cxyN            23 33-492-647-4972
+3.22400000000000E+001 HOUSEHOLD    quickly quiet packages
according to the fluffily ironic deposits cajo
        432 Customer#000000432        mnoCDJArKY0liRNkra7es1d3WdpRg6
23 33-307-912-9016   +5.71564000000000E+003 BUILDING     express, final
dolphins wake regular deposits. e
        830 Customer#000000830        XLdE4pELtz,H6i
23 33-408-548-6806   +7.77565000000000E+003 AUTOMOBILE   silent,
ironic requests sleep along the blithely regular instructions. final requests af
       1067 Customer#000001067        LYVoj1LJ Q                      23
33-764-123-9568   +9.15384000000000E+003 FURNITURE    final, bold
waters after the carefully silent ideas sleep even requests; fluffily unusual
requests across the care
      12874 Customer#000012874        ggrCZdqaBzgoFSVxa8qK
23 33-592-120-9012   +3.52163000000000E+003 HOUSEHOLD    slyly even
pinto beans mainta
      13037 Customer#000013037        rH ghadgknfLMG7U
23 33-821-137-5031   +2.53186000000000E+003 FURNITURE    finally
special foxes hang blithely deposits. final deposits nag blithely? ironic, ironic
excuses sleep careful
      13076 Customer#000013076        pvzvZ0Qax8xkDy
23 33-995-343-7109   +1.71650000000000E+002 BUILDING     slyly
pending asymptotes above the slyly special dugouts believe around the slyly
special dugouts. furious

  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
----------- ----------- ------------- ------------------------ ----------- ---------------
--------------- --------------
--------------------------------------------------------------------------------
     59718     7424651 F             +1.12649120000000E+005 01/01/1992  5-
LOW         Clerk#000269045             0 carefully regular pinto beans across
the even grouches dete
    334181     8600005 F             +9.40402300000000E+004 01/01/1992  2-
HIGH        Clerk#000061971             0 fluffily pending foxes haggle
carefully furiously final pinto beans. s
    360261    28122308 F             +1.49784400000000E+004 01/01/1992  5-
LOW         Clerk#000038000             0 express tithes doze stealthily around
the final requ
    368004    34593859 F             +9.69309000000000E+004 01/01/1992  2-
HIGH        Clerk#000193336             0 slyly unusual theodolites snooze
pending instructions! q
    414725    10491310 F             +2.65019800000000E+004 01/01/1992  5-
LOW         Clerk#000257604             0 sly accounts detect along the slyly
express
    466659    23364406 F             +1.09895870000000E+005 01/01/1992  4-
NOT SPECIFIED Clerk#000132830             0 blithely pending packages are.
carefully s
    470693    36896684 F             +2.39048960000000E+005 01/01/1992  3-
MEDIUM      Clerk#000040018             0 slyly final packages sleep fl
    560930     7513952 F             +2.58084620000000E+005 01/01/1992  3-
MEDIUM      Clerk#000005281             0 pinto beans use carefully quickly
ironic foxes! carefully ironic
    646854    40350565 F             +2.65661260000000E+005 01/01/1992  2-
HIGH        Clerk#000224020             0 furiously express requests boost
never across the slyly express pl
    682656    25155628 F             +1.90570620000000E+005 01/01/1992  5-
LOW         Clerk#000238603             0 silent ideas doubt along the careful

  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
----------- ----------- ----------- ----------- ---------- ----------
---------- ---------- ---------- ---------- ---------- ----------
---------- ---------- ---------- ----------
--------------------------------------------
    721220    53340877     1590929           2   +1.90000000000000E+001
+3.63889900000000E+004   +8.00000000000000E-002
+3.00000000000000E-002 R           F           01/02/1992 02/04/1992
01/09/1992    TAKE BACK RETURN          SHIP       regular packages caj
   1332613    16194306      444322           1   +1.40000000000000E+001
+1.95930000000000E+004   +8.00000000000000E-002
+7.00000000000000E-002 A           F           01/02/1992 02/11/1992
01/18/1992    TAKE BACK RETURN          TRUCK      furiously ironic foxes
sleep carefully
   3273383     7526949     1526950           3   +4.20000000000000E+001
+8.29739400000000E+004   +6.00000000000000E-002
+7.00000000000000E-002 R           F           01/02/1992 03/19/1992
01/18/1992    COLLECT COD               TRUCK      ironic asymptotes ac
   5885633    39766222     2266249           4   +2.70000000000000E+001
+3.47284800000000E+004   +0.00000000000000E+000
+5.00000000000000E-002 R           F           01/02/1992 03/10/1992
01/23/1992    COLLECT COD               TRUCK      even theodolites cajole
   6555845    29702527      452537           3   +2.80000000000000E+001
+4.27851200000000E+004   +1.00000000000000E-002
+5.00000000000000E-002 R           F           01/02/1992 03/11/1992
01/07/1992    NONE                      SHIP       carefully final foxes across the
silent fr
   7428384    27364021     1864040           4   +2.30000000000000E+001
+2.49241800000000E+004   +9.00000000000000E-002
+5.00000000000000E-002 R           F           01/02/1992 03/22/1992
01/03/1992    DELIVER IN PERSON         FOB        slyly express packages
are against the ex
  11174723    25531976     2281985           1   +4.10000000000000E+001
+8.22747000000000E+004   +7.00000000000000E-002
+2.00000000000000E-002 A           F           01/02/1992 03/28/1992
01/08/1992    NONE                      MAIL       pending, ironic theodolites are
fluffily
  12321157    18304491     1054498           4   +5.00000000000000E+001
+7.47290000000000E+004   +9.00000000000000E-002
+1.00000000000000E-002 R           F           01/02/1992 03/09/1992
01/10/1992    NONE                      SHIP       carefully ironic epitaphs haggle
  20713826    57220428     2470486           1   +1.00000000000000E+001
+1.34556000000000E+004   +9.00000000000000E-002
+8.00000000000000E-002 A           F           01/02/1992 03/11/1992
01/10/1992    TAKE BACK RETURN          AIR        pending accounts nag
blithely regular
  24447525     8854072      604075           4   +3.60000000000000E+001
+3.69226800000000E+004   +9.00000000000000E-002
+6.00000000000000E-002 R           F           01/02/1992 03/18/1992
01/18/1992    NONE                      AIR        blithely express packages around
the

  10 record(s) selected.
```

Query Substitution Variables

```
Power stream    Seed = 223142510
-- TPC TPC-H Parameter Substitution (Version 1.3.0)
-- using 223142510 as a seed to the RNG
Q1  DELTA       72
Q2  SIZE        9
```

```
   TYPE        BRASS
   REGION      EUROPE
Q3  SEGMENT     HOUSEHOLD
   DATE        1995-03-19
Q4  DATE        1995-12-01
Q5  REGION      ASIA
   DATE        1996-01-01
Q6  DATE        1996-01-01
   DISCOUNT    0.08
   QUANTITY    24
Q7  NATION1     VIETNAM
   NATION2     ALGERIA
Q8  NATION      ALGERIA
   REGION      AFRICA
   TYPE        PROMO BURNISHED NICKEL
Q9  COLOR       smoke
Q10 DATE        1993-05-01
Q11 NATION      UNITED STATES
   FRACTION    0.0000003333
Q12 SHIPMODE1   REG AIR
   SHIPMODE2   TRUCK
   DATE        1995-01-01
Q13 WORD1       unusual
   WORD2       requests
Q14 DATE        1995-03-01
Q15 DATE        1996-05-01
Q16 BRAND       Brand#13
   TYPE        ECONOMY ANODIZED
   SIZE1       38
   SIZE2       11
   SIZE3       42
   SIZE4       41
   SIZE5       43
   SIZE6       5
   SIZE7       12
   SIZE8       10
Q17 BRAND       Brand#34
   CONTAINER   JUMBO BAG
Q18 QUANTITY    314
Q19 BRAND1      Brand#42
   BRAND2      Brand#53
   BRAND3      Brand#55
   QUANTITY1   1
   QUANTITY2   16
   QUANTITY3   26
Q20 COLOUR      dark
   DATE        1997-01-01
   NATION      IRAN
Q21 NATION      SAUDI ARABIA
Q22 I1          21
   I2          10
   I3          32
   I4          20
   I5          23
   I6          17
   I7          26

Throughput Stream = 1    Seed = 223142511
-- TPC TPC-H Parameter Substitution (Version 1.3.0)
-- using 223142511 as a seed to the RNG
Q1  DELTA       80
Q2  SIZE        46
   TYPE        NICKEL
   REGION      AFRICA
Q3  SEGMENT     AUTOMOBILE
   DATE        1995-03-05
Q4  DATE        1993-09-01
Q5  REGION      EUROPE
   DATE        1996-01-01
Q6  DATE        1996-01-01
   DISCOUNT    0.06
   QUANTITY    25
Q7  NATION1     JORDAN
   NATION2     PERU
Q8  NATION      PERU
   REGION      AMERICA
   TYPE        ECONOMY BRUSHED BRASS
Q9  COLOR       salmon
Q10 DATE        1994-03-01
Q11 NATION      JAPAN
   FRACTION    0.0000003333
Q12 SHIPMODE1   SHIP
   SHIPMODE2   TRUCK
   DATE        1995-01-01
Q13 WORD1       unusual
   WORD2       requests
Q14 DATE        1995-07-01
Q15 DATE        1994-02-01
Q16 BRAND       Brand#54
   TYPE        STANDARD BURNISHED
   SIZE1       4
   SIZE2       7
   SIZE3       13
   SIZE4       10
   SIZE5       38
   SIZE6       34
   SIZE7       26
   SIZE8       2
Q17 BRAND       Brand#41
   CONTAINER   JUMBO PACK
Q18 QUANTITY    312
Q19 BRAND1      Brand#44
   BRAND2      Brand#31
   BRAND3      Brand#44
   QUANTITY1   6
   QUANTITY2   17
   QUANTITY3   22
Q20 COLOUR      orchid
   DATE        1995-01-01
   NATION      ALGERIA
Q21 NATION      JORDAN
Q22 I1          34
   I2          19
   I3          26
   I4          23
   I5          25
   I6          15
   I7          11

Throughput Stream = 2    Seed = 223142512
-- TPC TPC-H Parameter Substitution (Version 1.3.0)
-- using 223142512 as a seed to the RNG
Q1  DELTA       88
Q2  SIZE        34
   TYPE        COPPER
   REGION      EUROPE
Q3  SEGMENT     FURNITURE
   DATE        1995-03-21
Q4  DATE        1996-04-01
Q5  REGION      MIDDLE EAST
   DATE        1997-01-01
Q6  DATE        1997-01-01
   DISCOUNT    0.03
   QUANTITY    24
Q7  NATION1     ETHIOPIA
   NATION2     INDONESIA
Q8  NATION      INDONESIA
   REGION      ASIA
   TYPE        ECONOMY PLATED BRASS
Q9  COLOR       purple
Q10 DATE        1994-12-01
Q11 NATION      ALGERIA
   FRACTION    0.0000003333
Q12 SHIPMODE1   FOB
   SHIPMODE2   MAIL
   DATE        1995-01-01
Q13 WORD1       unusual
   WORD2       accounts
Q14 DATE        1995-10-01
Q15 DATE        1996-09-01
Q16 BRAND       Brand#34
   TYPE        MEDIUM POLISHED
   SIZE1       40
   SIZE2       2
   SIZE3       29
   SIZE4       7
   SIZE5       17
   SIZE6       33
   SIZE7       30
   SIZE8       10
Q17 BRAND       Brand#43
   CONTAINER   JUMBO DRUM
Q18 QUANTITY    313
Q19 BRAND1      Brand#41
```

```
   BRAND2       Brand#23
   BRAND3       Brand#43
   QUANTITY1    1
   QUANTITY2    18
   QUANTITY3    29
Q20 COLOUR       black
   DATE         1994-01-01
   NATION       KENYA
Q21 NATION       ETHIOPIA
Q22 I1           12
   I2           13
   I3           22
   I4           16
   I5           28
   I6           10
   I7           21

Throughput Stream = 3    Seed = 223142513
-- TPC TPC-H Parameter Substitution (Version 1.3.0)
-- using 223142513 as a seed to the RNG
Q1  DELTA        96
Q2  SIZE         22
   TYPE         STEEL
   REGION       AMERICA
Q3  SEGMENT      AUTOMOBILE
   DATE         1995-03-07
Q4  DATE         1994-01-01
Q5  REGION       AFRICA
   DATE         1997-01-01
Q6  DATE         1997-01-01
   DISCOUNT     0.09
   QUANTITY     24
Q7  NATION1      RUSSIA
   NATION2      ARGENTINA
Q8  NATION       ARGENTINA
   REGION       AMERICA
   TYPE         ECONOMY ANODIZED BRASS
Q9  COLOR        peach
Q10 DATE         1993-09-01
Q11 NATION       JORDAN
   FRACTION     0.0000003333
Q12 SHIPMODE1    TRUCK
   SHIPMODE2    MAIL
   DATE         1995-01-01
Q13 WORD1        unusual
   WORD2        accounts
Q14 DATE         1996-01-01
Q15 DATE         1994-06-01
Q16 BRAND        Brand#14
   TYPE         ECONOMY BRUSHED
   SIZE1        23
   SIZE2        16
   SIZE3        45
   SIZE4        11
   SIZE5        21
   SIZE6        19
   SIZE7        37
   SIZE8        42
Q17 BRAND        Brand#45
   CONTAINER    WRAP BAG
Q18 QUANTITY     315
Q19 BRAND1       Brand#53
   BRAND2       Brand#51
   BRAND3       Brand#42
   QUANTITY1    7
   QUANTITY2    19
   QUANTITY3    26
Q20 COLOUR       light
   DATE         1997-01-01
   NATION       EGYPT
Q21 NATION       RUSSIA
Q22 I1           16
   I2           20
   I3           26
   I4           11
   I5           30
   I6           10
   I7           32

Throughput Stream = 4    Seed = 223142514
-- TPC TPC-H Parameter Substitution (Version 1.3.0)
-- using 223142514 as a seed to the RNG
Q1  DELTA        104
Q2  SIZE         10
   TYPE         BRASS
   REGION       EUROPE
Q3  SEGMENT      FURNITURE
   DATE         1995-03-23
Q4  DATE         1996-08-01
Q5  REGION       AMERICA
   DATE         1997-01-01
Q6  DATE         1997-01-01
   DISCOUNT     0.06
   QUANTITY     25
Q7  NATION1      KENYA
   NATION2      CHINA
Q8  NATION       CHINA
   REGION       ASIA
   TYPE         LARGE POLISHED BRASS
Q9  COLOR        navy
Q10 DATE         1994-06-01
Q11 NATION       ARGENTINA
   FRACTION     0.0000003333
Q12 SHIPMODE1    RAIL
   SHIPMODE2    MAIL
   DATE         1996-01-01
Q13 WORD1        unusual
   WORD2        accounts
Q14 DATE         1996-04-01
Q15 DATE         1996-12-01
Q16 BRAND        Brand#54
   TYPE         SMALL BURNISHED
   SIZE1        47
   SIZE2        6
   SIZE3        30
   SIZE4        27
   SIZE5        28
   SIZE6        43
   SIZE7        7
   SIZE8        1
Q17 BRAND        Brand#42
   CONTAINER    WRAP PACK
Q18 QUANTITY     312
Q19 BRAND1       Brand#55
   BRAND2       Brand#34
   BRAND3       Brand#32
   QUANTITY1    2
   QUANTITY2    20
   QUANTITY3    22
Q20 COLOUR       spring
   DATE         1995-01-01
   NATION       ROMANIA
Q21 NATION       KENYA
Q22 I1           16
   I2           23
   I3           15
   I4           13
   I5           12
   I6           22
   I7           17

Throughput Stream = 5    Seed = 223142515
-- TPC TPC-H Parameter Substitution (Version 1.3.0)
-- using 223142515 as a seed to the RNG
Q1  DELTA        112
Q2  SIZE         48
   TYPE         NICKEL
   REGION       AMERICA
Q3  SEGMENT      MACHINERY
   DATE         1995-03-09
Q4  DATE         1994-05-01
Q5  REGION       ASIA
   DATE         1997-01-01
Q6  DATE         1997-01-01
   DISCOUNT     0.04
   QUANTITY     24
Q7  NATION1      FRANCE
   NATION2      IRAN
Q8  NATION       IRAN
   REGION       MIDDLE EAST
   TYPE         LARGE BURNISHED BRASS
Q9  COLOR        metallic
Q10 DATE         1993-03-01
Q11 NATION       KENYA
```

FRACTION 0.0000003333
Q12 SHIPMODE1 AIR
SHIPMODE2 MAIL
DATE 1996-01-01
Q13 WORD1 unusual
WORD2 accounts
Q14 DATE 1996-07-01
Q15 DATE 1994-09-01
Q16 BRAND Brand#34
TYPE LARGE PLATED
SIZE1 8
SIZE2 12
SIZE3 6
SIZE4 3
SIZE5 25
SIZE6 14
SIZE7 20
SIZE8 50
Q17 BRAND Brand#43
CONTAINER WRAP DRUM
Q18 QUANTITY 314
Q19 BRAND1 Brand#52
BRAND2 Brand#22
BRAND3 Brand#31
QUANTITY1 7
QUANTITY2 10
QUANTITY3 29
Q20 COLOUR frosted
DATE 1994-01-01
NATION INDONESIA
Q21 NATION FRANCE
Q22 I1 14
I2 20
I3 22
I4 18
I5 19
I6 23
I7 28

Throughput Stream = 6 Seed = 223142516
-- TPC TPC-H Parameter Substitution (Version 1.3.0)
-- using 223142516 as a seed to the RNG
Q1 DELTA 120
Q2 SIZE 35
TYPE TIN
REGION EUROPE
Q3 SEGMENT FURNITURE
DATE 1995-03-25
Q4 DATE 1996-12-01
Q5 REGION EUROPE
DATE 1993-01-01
Q6 DATE 1993-01-01
DISCOUNT 0.09
QUANTITY 24
Q7 NATION1 UNITED KINGDOM
NATION2 BRAZIL
Q8 NATION BRAZIL
REGION AMERICA
TYPE MEDIUM BRUSHED STEEL
Q9 COLOR light
Q10 DATE 1994-01-01
Q11 NATION BRAZIL
FRACTION 0.0000003333
Q12 SHIPMODE1 REG AIR
SHIPMODE2 FOB
DATE 1996-01-01
Q13 WORD1 unusual
WORD2 deposits
Q14 DATE 1996-11-01
Q15 DATE 1997-04-01
Q16 BRAND Brand#14
TYPE PROMO BRUSHED
SIZE1 35
SIZE2 37
SIZE3 24
SIZE4 31
SIZE5 21
SIZE6 43
SIZE7 42
SIZE8 7
Q17 BRAND Brand#45
CONTAINER SM BAG
Q18 QUANTITY 315
Q19 BRAND1 Brand#14
BRAND2 Brand#55
BRAND3 Brand#35
QUANTITY1 2
QUANTITY2 11
QUANTITY3 25
Q20 COLOUR puff
DATE 1997-01-01
NATION UNITED KINGDOM
Q21 NATION UNITED KINGDOM
Q22 I1 21
I2 30
I3 13
I4 16
I5 12
I6 22
I7 19

Appendix D: Driver Source Code

setupRun

```
#!/usr/bin/perl
# usage  setupRun
# This shell script will set up a file with the variables necessary for each run
# into the tpcd.runsetup file
# If sets the timestamp of the output directory, and creates some of the output
# directories.

# Get TPC-D specific environment variables
require 'getvars';
require "macro.pl";

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

#verify environment variables required for script to run
@reqVars     = ("TPCD_AUDIT_DIR",
                "TPCD_DBNAME",
                "TPCD_PATH_DELIM",
                "TPCD_SF",
                "TPCD_NUMSTREAM",
                "TPCD_QUERY_TEMPLATE_DIR",
                "TPCD_GEN_UPDATEPAIRS",
                "TPCD_PRODUCT",
                "TPCD_AUDIT");

&setVar(@reqVars, "ERROR");


# set up local vars
$auditDir=          $ENV{"TPCD_AUDIT_DIR"};
$dbname= $ENV{"TPCD_DBNAME"};
$delim=             $ENV{"TPCD_PATH_DELIM"};
$sf=                $ENV{"TPCD_SF"};
$numStream=         $ENV{"TPCD_NUMSTREAM"};
$templt_dir=        $ENV{"TPCD_QUERY_TEMPLATE_DIR"};
$updatePairs=       $ENV{"TPCD_GEN_UPDATEPAIRS"};
$product= $ENV{"TPCD_PRODUCT"};
$RealAudit=         $ENV{"TPCD_AUDIT"};

# if runnumber is not set, then set it to 0 and run the script that must
# be run on a one time basis, but after the db has been created
if (length($ENV{"TPCD_RUNNUMBER"}) <= 0){
  $runNum=0;
}
else{
  $runNum=$ENV{"TPCD_RUNNUMBER"};
}

# increment the run number and use this to name the directory
$runNum++;
$tpcdTimestamp=`perl gettimestamp "short"`;
$runDir="$auditDir${delim}auditruns${delim}run$runNum";

#remove the old tpcd.runsetup file and recreate it with the new values
if ( -e "tpcd.runsetup" ){
  &rm("tpcd.runsetup");
}

open(XX,">tpcd.runsetup");
#set the environment variables into the tpcd.runsetup file
print XX "TPCD_RUNNUMBER=$runNum          # current run number\n";
print XX "TPCD_TIMESTAMP=$tpcdTimestamp   # timestamp for this
run\n";
print XX "TPCD_RUN_DIR=$runDir            # current output directory for
run\n";
close(XX);

mkdir ($runDir, 0775) || die "cannot mkdir $runDir";
mkdir ("$runDir${delim}querytext", 0775) || die "cannot mkdir $runDir$
{delim}querytext";

# generate the power and throughput test query files for this run
# this generates the appropriate sql files for each stream in the power
# and throughput tests of this run
$ENV{"TPCD_RUNNUMBER"}=$runNum;
$ENV{"TPCD_TIMESTAMP"}=$tpcdTimestamp;
$ENV{"TPCD_RUN_DIR"}=$runDir;

# build the query run text that are specific to this run
#require "buildqueryruntext";
# build the queries for the test database ( provided scale factor )
print "About to build the queries for the test database ...\n";
#print "NOTE: You will see 2 messages saying No variable definitions...\n";
#print "      This is okay to see as the update functions do not have \n ";
#print "      substitution parameters\n";

$rc = 0;
$rc = system("perl buildpow.test $sf $templt_dir");
if ( $rc != 0 )
{
   die "building queries for power run failed buildpow.test rc=$rc\n";
}

# build the queries for the throughput test ( provided scale factor
# and number of streams )

$rc = system("perl buildthru $sf $numStream $templt_dir");
if ( $rc != 0 )
{
   die "building queries for throughput run failed buildthru rc=$rc\n";
}


# record what database we are using....just for information sake
system("echo database name = $dbname > $runDir${delim}
dbrun$runNum.info");
`db2 list database directory >> $runDir${delim}dbrun$runNum.info`;
`db2 terminate`;

# reminder to "clean" system before the real runs
print "Ensure that a full cleanup is done of the system via a reboot before\n";
print "the power and throughput tests are run\n";
print "For internal testing purposes, db2stop or db2start can be used
instead.\n";
print "Run number is: $runNum\n";

system("ln -sf $runDir $auditDir${delim}auditruns${delim}CURRUN");
if (@ARGV > 0)
{
 $outfile="$auditDir${delim}auditruns${delim}CURRUN/README.run$
{runNum}";
 open(fs,">${outfile}") || die "Can't open ${outfile}: $!\n";
 print fs "@ARGV\n";
 close(fs);
}

1;
```

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($ENV{"TPCD_ROOTPRIV"}) <= 0)
{
 die "TPCD_ROOTPRIV environment variable not set - yes/no \n";
}
if (length($ENV{"TPCD_STATS_INTERVAL"}) <= 0)
{
 die "TPCD_STATS_INTERVAL environment variable not set - interval in
seconds \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"};
$stats_int=$ENV{"TPCD_STATS_INTERVAL"};
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 runthroughtput
tests
  system("echo \"The following disks are assigned to the indicated volume
groups\" > $runDir/sysunused$runNum") && die "cannot create
$runDir/sysunused$runNum";
```

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

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

&getConfig("p");
if ( $rootPriv eq "yes" )
{
   # get the o/s tuning parameters...currently AIX only and only if your
   # user has root privileges to run this
   &getOSTune("p");
}
if ($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 succesfully.\n";
}
else
{
   print "Power stream failed. ret=$ret\n";
}

if ($platform eq "aix")
{
   # show that the same disks are still used or unused
   # system("getdisks \"After completion of the Power Test\"");

   #clean up
}
if ( $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")
      {
             $rc= `$prefix perl5 $auditDir${delim}tools${delim}zap $a"-f$a"
$a"^sar$a" `;
      $rc= `$prefix perl5 $auditDir${delim}tools${delim}zap $a"-f$a"
$a"^sadc$a" `;
      }
     else
      {
      $rc= `$prefix perl5 $auditDir${delim}tools${delim}zap $a"-f$a"
$a"^vmstat$a" `;
      $rc= `$prefix perl5 $auditDir${delim}tools${delim}zap $a"-f$a"
$a"^iostat$a" `;
      $rc= `$prefix perl5 $auditDir${delim}tools${delim}zap $a"-f$a"
$a"pmlogger$a" `;
             $rc= `$prefix perl5 $auditDir${delim}tools${delim}zap $a"-f$a"
$a"^sar$a" `;
      $rc= `$prefix perl5 $auditDir${delim}tools${delim}zap $a"-f$a"
$a"^sadc$a" `;
      system("killall sadc sar pmlogger vmstat");

             #
             # doug
             #
      `pcp_gif_maker.csh "$runDir" "$stats_int"`;
             `pcp_log_cruncher.csh "$runDir" "$stats_int"`;
      system("ps -Alf >> $runDir${delim}ps_after.txt");

             #
             # process sar binary into desired text output
             #
             #
      $sar_bin_name = `ls $runDir${delim}*sar_binary`;
      print ("sar_bin_name is: $sar_bin_name");

      system("/mnt/disk2/tpch/tpcd/tools/process_sar.csh $runDir $stats_int");

             system("/mnt/disk2/tpch/tpcd/tools/run_grab_proc >> $runDir$
{delim}dbrun${runNum}.proc.after");

      $rc= `$prefix iostat$suffix `;

      }
     $rc= `$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 dependancy that this mpstrcnt file exist before the
# calcmetrics.pl script is called, both because it is used as input for
# calcmetrics.pl, and because the output from calcmetrics is used as
# the trigger for watchstreams to complete, and watchstreams cats its
# output at the end of the mstrcnt file.

# generate the mpinter?.metrics file in the run directory
#require 'calcmetricsp.pl';
if ( $runUF eq "UF")
{
  system("perl calcmetricsp.pl UF");
}
else
{
  system("perl calcmetricsp.pl");
}

# concatenate all the throughput inter files that were used to
# generate these results into the calcmetrics output file (mpinterX.metrics)
#cd $TPCD_RUN_DIR
&cat("$runDir${delim}mpqinter*","$runDir${delim}
mpinter$runNum.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_all db2 get database configuration for $dbname >>
$dbtunefile");
  system("db2 get database manager configuration >> $dbtunefile");
  system("db2set >> $dbtunefile");
  if (( $mode eq "mln" ) || ( $mode eq "mpp"))
  {
    $cfgfile="$runDir${delim}dbtune${runNum}.";
    #removed by Alex due to hang
    #system("db2_all '||\" typeset -i ln=##; db2 get db cfg for $dbname >
$cfgfile\${ln} ; db2 get dbm cfg >> $cfgfile\${ln}; db2set >> $cfgfile\${ln};
db2 terminate '");
  }

}

sub getOSTune
{
  $testtype=$_[0];
  if ( $platform eq "aix" )
  {
     print "Getting OS and VMdatabase configuration.\n";
     $ostunefile="$runDir${delim}m${testtype}ostune${runNum}";
     open(OSTUNE, ">$ostunefile") || die "Can't open $ostunefile: $!\n";
     $timestamp=`perl gettimestamp "long"`;
     print OSTUNE "Operating System and Virtual Memory configuration
taken at : $timestamp";
     close(OSTUNE);
     system("${delim}usr${delim}samples${delim}kernel${delim}schedtune
>> $ostunefile");
     system("${delim}usr${delim}samples${delim}kernel${delim}vmtune >>
$ostunefile");
  }
  else
  {
     print "OS parameters retrieval not supported for $platform \n";
  }

}

sub verifyTPCDbatch
{
  $logfile=$_[0];
  $dbname=$_[1];
  $file="verifytpcdbatch.clp";
  open(VERTBL, ">$file") || die "Can't open $file: $!\n";
  print VERTBL "connect to $dbname;\n";
  print VERTBL "select name,creator,valid,last_bind_time,isolation from
sysibm.sysplan where name like 'TPCD%';\n";
  print VERTBL "connect reset;\n";
  print VERTBL "terminate;\n";
  close(VERTBL);
  system("db2 -vtf $file >> $logfile");
}
#
```

nthroughput

```
:   # -*-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"};

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

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

# return 1 if the given pattern(parameter $_[0]) matches any file
sub existfile {
  if ($platform eq "aix" || $platform eq "sun" || $platform eq "ptx" || $platform
eq "linux")
  {
    `ls $_[0] 2> /dev/null | wc -l` + 0 != 0;
  }
  else
  {
    `dir /b $_[0] 2> NUL | wc -l` + 0 != 0;
  }
}

if ($inlistmax eq "default")
{
  $inlistmax = 400;
}

# no longer stop and start the dbm between runs when not in realaudit mode
#if ( $RealAudit ne "yes" )
#{
#    # if we are not in real audit mode then we must start the db manager now
#    system("db2start");
#    # activate the database
#    system("db2 activate database $dbname");
#}

$misofile="$runDir${delim}miso$runNum";
# append isolation level information about tpcdbatch to the miso file
open(MISO, ">>$misofile") || die "Can't open $misofile: $!\n";
$curTs = `perl gettimestamp "long"`;
print MISO "Timestamp and isolation level of tpcdbatch before throughput run
at : $curTs\n";
close(MISO);

if ( $product eq "pe" )
{
   system("db2 \"connect to $dbname\"; db2 \"select
name,creator,valid,unique_id,isolation from sysibm.sysplan where name like
'TPCD%'\" >> $runDir${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 t -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")
      {
         $rc= `perl5 zap "-f" "sar"`;
         $rc= `perl5 zap "-f" "sadc"`;
      }
      else
      {
         $rc= `perl5 zap "-f" "vmstat"`;
         $rc= `perl5 zap "-f" "iostat"`;
               system("killall sadc sar pmlogger vmstat");
      }
      if ( $pn > 1 )
      {
         $rc= `perl5 zap "-f" "netstat"`;
      }
      $rc= `perl5 zap "-f" "getstats"`;
   }
}

open(MISO, ">>$misofile") || die "Can't open $misofile: $!\n";
$curTs = `perl gettimestamp "long"`;
print MISO "Timestamp and isolation level of tpcdbatch after throughput run
at : $curTs\n";
close(MISO);

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

if ( $RealAudit ne "yes" )
{
   $curTs = `perl gettimestamp "short"`;
   # grab the db and dbm snapshot before we deactivate
   system("db2 get snapshot for all on $dbname > $runDir${delim}
dbTrun$runNum.snap.$curTs");
   system("db2 get snapshot for database manager >> $runDir${delim}
dbTrun$runNum.snap.$curTs");
}

# now copy the reports from the count of streams files into one final file
&cat("$runDir${delim}strcnt*","$runDir${delim}mstrcnt$runNum");
#(NOTE: there is a dependancy that this mstrcnt file exist before the
# calcmetrics.pl script is called, both because it is used as input for
# calcmetrics.pl, and because the output from calcmetrics is used as
# the trigger for watchstreams to complete, and watchstreams cats its
# output at the end of the mstrcnt file.

# generate the mtinter?.metrics file in the run directory
#require 'calcmetrics.pl';

if ( $runUF ne "no")
{
  system("perl calcmetrics.pl $numStream UF");
}
else
{
  system("perl calcmetrics.pl $numStream");
}

# concatenate all the throughput inter files that were used to
# generate these results into the calcmetrics output file (mtinterX.metrics)
#cd $TPCD_RUN_DIR
&cat("$runDir${delim}mts*inter*","$runDir${delim}
mtinter$runNum.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");
$rc=&dodb_noconn("db2 get db cfg for $dbname | grep -i log >> $runDir$
{delim}endLog.Info",$all_ln);
if ( $logDir ne "NULL" )
{
   $rc=&dodb_noconn("$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
$rc = system("perl catinfo.pl p");


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

#Report current log info to the run# directory in a file called endLog.Info
system("perl getLogInfo.pl endLog");

# if we are in audit mode we must do a db2stop at the end of the
power/throughput run
#if ( $RealAudit eq "yes" )
#{
   #system("db2stop");
#}

1;

sub getConfig
{
  $testtype=$_[0];
  print "Getting database configuration.\n";
  $dbtunefile="$runDir${delim}m${testtype}dbtune${runNum}";
  open(DBTUNE, ">$dbtunefile") || die "Can't open $dbtunefile: $!\n";
  $timestamp=`perl gettimestamp "long"`;
  print DBTUNE "Database and Database manager configuration taken at :
$timestamp";
  close(DBTUNE);
  system("db2level >> $dbtunefile");
  system("db2 get database configuration for $dbname >> $dbtunefile");
  system("db2 get database manager configuration >> $dbtunefile");
  system("db2set >> $dbtunefile");

}

sub getOSTune
{
  $testtype=$_[0];
  if ( $platform eq "aix" || $platform eq "linux")
  {
     print "Getting OS and VMdatabase configuration.\n";
     $ostunefile="$runDir${delim}m${testtype}ostune${runNum}";
     open(OSTUNE, ">$ostunefile") || die "Can't open $ostunefile: $!\n";
     $timestamp=`perl gettimestamp "long"`;
     print OSTUNE "Operating System and Virtual Memory configuration
taken at : $timestamp";
     close(OSTUNE);
     system("${delim}usr${delim}samples${delim}kernel${delim}schedtune
>> $ostunefile");
     system("${delim}usr${delim}samples${delim}kernel${delim}vmtune >>
$ostunefile");
  }
  else
  {
     print "OS parameters retrieval not supported for $platform \n";
  }

}

sub verifyTPCDbatch
{
  $logfile=$_[0];
  $dbname=$_[1];
  $file="verifytpcdbatch.clp";
  open(VERTBL, ">$file") || die "Can't open $file: $!\n";
  print VERTBL "connect to $dbname;\n";
  print VERTBL "select name,creator,valid,last_bind_time,isolation from
sysibm.sysplan where name like 'TPCD%';\n";
  print VERTBL "connect reset;\n";
  print VERTBL "terminate;\n";
  close(VERTBL);
  system("db2 -vtf $file >> $logfile");
}
```

Makefile.linux

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

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;

#ifndef SQLWINT
  int              su_semid;
  key_t            su_semkey;
#else
  HANDLE           su_hSem;
  char             UF1_semfile[256];
#endif

  char myoutstreamfile[256];
  FILE *myoutstream;

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

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

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


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

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


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

  /* Start processing each chunk in my range */
#ifdef UF1DEBUG
  fprintf (myoutstream,"Before loop_a startChunk = %d, stopChunk = %d\n",
startChunk, stopChunk);
  fflush(myoutstream);
#endif
  for ( UF_chunk = startChunk; UF_chunk <= stopChunk; UF_chunk++ )
/*jenCI*/
  {                                                                /*jenCI*/
    /* wlc  062797 */
    sqlcode = SQL_RC_E911;
    month = (short)UF_chunk;  /* Cast 'short' added bbe */
    maxwait = 1;
    rc = 0;

#ifdef UF1DEBUG
  fprintf (myoutstream, "Before While_a Chunk= %d \n",UF_chunk);
  fflush(myoutstream);
#endif
    /* Loop to handle any deadlocks */
    while (sqlcode == SQL_RC_E911 && maxwait <= MAXWAIT &&
rc==0)
    {
```

```
        sqlcode = 0;
#ifdef UF1DEBUG
  fprintf (myoutstream, "in loop before orders exec sql\n");
  fflush(myoutstream);
#endif
        EXEC SQL INSERT INTO TPCD.ORDERS
              SELECT
O_ORDERKEY,O_CUSTKEY,O_ORDERSTATUS,O_TOTALPRICE,
                    O_ORDERDATE,O_ORDERPRIORITY,O_CLERK,O_SHIPP
RIORITY,O_COMMENT
              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 (myoutstream,
                  "\nDeadlock detected inserting from tpcdtemp.orders_new for
chunk %d for pair %d..Retrying...\n",UF_chunk,updatePair);
          SleepSome(UF_DEADLOCK_SLEEP);
          maxwait++;                          /* jen DEADLOCK */
        }
        else if (sqlcode < 0)
        {
          fprintf(myoutstream,
                "Insert into orders pair %d chunk %d failed sqlcode=%d\n",
                updatePair,UF_chunk,sqlcode);
          dumpCa(&sqlca);
          rc = -1;
        }
        else
        {
          /* Everything worked with ORDERS, proceed with LINEITEM */
          saveInsertedOrders = sqlca.sqlerrd[2];

          sqlcode = 0;
#ifdef UF1DEBUG
          fprintf (myoutstream, "in lineitem for update pair number %d set %d
chunk %d\n",
                  updatePair, i,UF_chunk);
          fflush(myoutstream);
#endif

          EXEC SQL INSERT INTO TPCD.LINEITEM
            SELECT
L_ORDERKEY,L_PARTKEY,L_SUPPKEY,L_LINENUMBER,L_QUANTIT
Y,
            L_EXTENDEDPRICE,L_DISCOUNT,L_TAX,
            L_RETURNFLAG,L_LINESTATUS,L_SHIPDATE,L_COMMITDA
TE,L_RECEIPTDATE,
            L_SHIPINSTRUCT,L_SHIPMODE,L_COMMENT
            FROM TPCDTEMP.LINEITEM_NEW WHERE APP_ID = :
UF_chunk;
            /*(AND L_ORDERKEY IN
              (SELECT O_ORDERKEY FROM TPCD.ORDERS
                WHERE 12*(YEAR(O_ORDERDATE)-1992)+MONTH
(O_ORDERDATE)-01 = :month);*/

          if (sqlca.sqlcode < 0)
            sqlcode = error_check();

          if (sqlcode == SQL_RC_E911)
          {         /* we've hit a deadlock */
            fprintf (myoutstream,
                    "\nA deadlock has been detected inserting from
tpcdtemp.lineitem%d_%d...Retrying...\n",
                    updatePair, UF_chunk);
            SleepSome(UF_DEADLOCK_SLEEP);
            maxwait++;                        /* jen DEADLOCK */
          }
          else if (sqlcode < 0)
          {
            fprintf(myoutstream,
                  "Insert into lineitem pair %d chunk %d failed sqlcode=%d\n",
                  updatePair,UF_chunk,sqlcode);
            dumpCa(&sqlca);
            rc = -1;
          }
          else
          {
#ifdef UF1DEBUG
            fprintf (myoutstream, "lineitem insert succeeded\n");
            fflush(myoutstream);
#endif
            /* accumulate the number of row inserted */
            /* Order count ONLY updated if both orders and lineitem */
            /* go through */
            insertedOrders += saveInsertedOrders;              /* kbs */
            insertedLineitem += sqlca.sqlerrd[2];
            rc=0;
            EXEC SQL COMMIT WORK;
            error_check();

#ifdef UF1DEBUG
            /* report the number of row inserted */
            fprintf(myoutstream, "    interim %ld rows for chunk %d into
TPCD.ORDERS at %*.*s\n",
                    insertedOrders,UF_chunk,T_STAMP_1LEN,T_STAMP_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(myoutstream,
                    "    interim %ld rows for chunk %d into TPCD.LINEITEM at %
*.*s\n",
                    insertedLineitem,UF_chunk,
                    T_STAMP_1LEN,T_STAMP_1LEN,  /* TIME_ACC jen*/
                    get_time_stamp(T_STAMP_FORM_1,
                            (Timer_struct *)NULL));  /* TIME_ACC jen*/

            fprintf( myoutstream,
                    "    inserts for update pair %d chunk %d complete at %
*.*s\n\n",
                    updatePair, UF_chunk,
                    T_STAMP_1LEN,T_STAMP_1LEN,  /* TIME_ACC jen*/
                    get_time_stamp(T_STAMP_FORM_1,
                            (Timer_struct *)NULL));  /* TIME_ACC jen*/

#endif
          }
        } /* process lineitem INSERTs */
      } /* while loop for deadlocks */
    } /* while processing chunks */

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

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

      goto UF1_exit;
    }

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


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

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

```
#ifndef SQLWINT
  /*    we will use the tpcd.setup file to generate the semaphore key   begin
SEMA */
  if (getenv("TPCD_AUDIT_DIR") != NULL)
  {
    /* this is assuming that you will be running this from 0th node */
    sprintf(sourcefile, "%s%ctools%ctpcd.setup",
          getenv("TPCD_AUDIT_DIR"), PATH_DELIM,PATH_DELIM);
  }
  else
  {
    fprintf (stderr, "Can't open UF1 semaphore file TPCD_AUDIT_DIR is not
defined.\n");
    exit (-1);
  }
  /* end SEMA */

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

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

    sleep(1);
  }

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



/*****************************************************************
*****/
/* UF2 child                                          */
/*****************************************************************
*****/
void runUF2_fn ( int updatePair, int thisConcurrentDelete, int numChunks,
char *dbname, char *userid, char *passwd )
{
  int rc = 0;
  long sqlcode;
  int maxwait;
  int startChunk = thisConcurrentDelete*numChunks; /* where do we start? */
  long 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_semfile[256];
#endif

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

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


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

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

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

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

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

#ifdef UF2DEBUG
  fprintf (myoutstream, "after connect  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(myoutstream, "    interim %ld rows for chunk %d from
TPCD.ORDERS at %*.*s\n",
                deletedOrders,UF_chunk,T_STAMP_1LEN,T_STAMP_1LEN,
/* TIME_ACC jen*/
                get_time_stamp(T_STAMP_FORM_1,(Timer_struct *)NULL));
/* TIME_ACC jen*/
          fprintf(myoutstream, "    interim %ld rows for chunk %d from
TPCD.LINEITEM at %*.*s\n",
                deletedLineitems,UF_chunk,T_STAMP_1LEN,T_STAMP_1LE
N,  /* TIME_ACC jen*/
                get_time_stamp(T_STAMP_FORM_1,(Timer_struct *)NULL));
/* TIME_ACC jen*/
          fprintf( myoutstream,
                "    deletes for update pair %d chunk %d complete at %
*.*s\n\n",
                updatePair, UF_chunk,
                T_STAMP_1LEN,T_STAMP_1LEN,  /* TIME_ACC jen*/
                get_time_stamp(T_STAMP_FORM_1,
                        (Timer_struct *)NULL));  /* TIME_ACC jen*/
#endif
        }
      } /* process orders  deletes */
    } /* while trying to delete one chunk loop */
  } /* while there are more chunks */

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

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

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

```
UF2_exit:
  fclose (myoutstream);

  /* exiting, increment the semaphore */
#ifndef 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 semaphone UF2 set %d\n",
thisConcurrentDelete);
    exit(1);
  }                                                  /*jenSEM*/

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

    SleepSome(1);
  }

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

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

#
```

tpcdbatch.h

```
/*****************************************************************
****************
*
*   TPCDBATCH.H
*
*  Revision History:
*
*  27 may 99 bbe from (24 nov 98 jen) fixNTtimestamp - fixed NT timestamp
to print millisecond correctly
*  27 may 99 bbe from (10 dec 98 jen) SUN - added Haider's changes
necessary for SUN
*  17 jun 99 jen  Increased version to 5.1
*  10 aug 99 bbe  Increased version to 5.2
*  13 aug 99 bbe  Increased version to 5.3
*  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 tjg */
#define TPCDBATCH_SELECT      20
#define TPCDBATCH_NONSELECT   30
#define TPCDBATCH_EOBLOCK     40                    /* @d30369 tjg */
#define TPCDBATCH_INSERT      50
#define TPCDBATCH_DELETE      60

#define TPCDBATCH_MAX_COLS    100                   /* @d30369 tjg
*/

#define TPCDBATCH_CHAR char
```

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

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

#define PARALLEL_UPDATES 1

#ifdef PARALLEL_UPDATES
#define UF1OUTSTREAMPATTERN "%s%cuf1.%02d.%d.out"
#ifdef TPCD_NONPARTITIONED
#define UF2OUTSTREAMPATTERN "%s%cuf2.%02d.%d.out"
#else
/* kelly add same as NONPART. */
#define UF2OUTSTREAMPATTERN "%s%cuf2.%02d.%d.out"
/*  kelly ... take this out ... should be same name as for non-paritioned
#define UF2OUTSTREAMPATTERN "%s%cuf2.%02d.%d.%d.out"   */     /
*DELjen add delchunk*/
#endif
#define BUFSIZE 1024
#endif


#define T_STAMP_FORM_1 1
#define T_STAMP_FORM_2 2
/* jen TIME_ACC start */
#define T_STAMP_FORM_3 3
#define T_STAMP_1LEN 17
#if defined (SQLUNIX) || defined (SQLAIX) || 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    "r\0"
#define WRITEMODE   "w\0"
#define APPENDMODE  "a\0"
#define mem_error(xx)                                   \
{ fprintf(stderr,"\n--Out of memory when %s.\n",xx); }
/* Display out-of-memory and end */

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



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

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

/** Timer-necessary defines for portability **/
#if (defined (SQLOS2) || defined(SQLWINT)) || defined(SQLWIN) || defined
(SQLDOS)
typedef struct timeb Timer_struct;
#elif (defined (SQLUNIX) || defined(SQLAIX) || 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 mseedme file wasn't getting read
*              correctly - EOF processing done too soon.
*
*  22 Feb 96 kbs  port to NT
*  26 Mar 96 kbs  Fix to avoid countig UFs as queries for min max
*  27 Jun 97 wlc Temporarily fixed deadlock problems when doing UF1, UF2
*  30 Jul 97 wlc Add in support for load_update and TPCD_SPLIT_DELETES
*  13 Aug 97 wlc fixed UF1 log file formatting problem,
*               using TPCD_TMP_DIR for temp files instead of /tmp,
*               make summary table fit in 80-column,
*               fixed UF2 # of deleted rows reporting problem
*  18 Aug 97 wlc added command line support for inlistmax
*  20 Aug 97 wlc added support for runthroughput without UF
*  27 Aug 97 aph Replaced hardcoded 'tpcdaudit' with getenv
("TPCD_AUDIT_DIR")
*  05 Sep 97 wlc fixing free() problem in NT
*  26 Sep 97 kmw change FLOAT processing in echo_sqlda and
print_headings
*  10 oct 97 jen  add lock table in share mode for staging tables
*  21 oct 97 jen added explicit rollback on failure of uf1
*  27 oct 97 jen don't update TPCD.xxxx.update.pair.num if not running UFs
in
*              throughput run
*  01 nov 97 jen temp code to do a prep then execute stmt in UFs so we can
*              get timings
*  03 nov 97 jen realligned UF code for readablility
*              pushed UF2 commit into loop for inlistmax
*              fixed UF2 code so rollback performed
*  04 nov 97 jen Added code to handle vldb
*  06 nov 97 jen Commented out temp code for prep then execute stmts using
*              TPCD_PREPARETIME def
*              Updated version number to 2.2
*              send all output during update functiosn to output files, not
*              stderr
*  10 nov 97 jen jenCI Updated version number to 2.3
*              Added handling of TPCD_CONCURRENT_INSERTS. Change
control of
*              chunk processing to use the concurrent_inserts value as the
*              control.  Now the inserts will be run in
TPCD_CONCURRENT_INSERTS
/ *            sets, each having concurrent_inserts/
```

```
*  13 nov 97 jen jen DEADLOCK.  FIxed bug that Alex found where deadlock for tpcd.setup
count
*            (maxwait) was incremented on every execution of the stmt as
*            opposed to just when deadlock really happened.
*  14 nov 97 jen jenSEM - fix up error reporting on semaphore failure
*            sem_op now returns failure to caller so caller can report where
*            failure has happened.
*            Forced dbname to be upper case, an all other parts of update
*            pair number to be lowercase
*  15 nov 97 jen SEED Reworked code to grab the seed from the seed file.
Now
*            reusing seeds between runs, so power run will always use first
*            seed, throughput will use the 2nd - #stream+1 seeds
*
*  13 jan 98 jen LONG  Increase stmt_str to be able to hold inlists with larger
*            order key numbers
*  04 mar 98 jen IMPORT added support for TPCD_UPDATE_IMPORT to
chose whether
*            using import or load api's for loading data into the staging
*            tables
*  04 mar 98 jen TIMER changed from using gettimer to gettimeofday for unix
*  01 apr 98 jen Fixed IMPORT code to do the proper checking on strcmp (ie !
strcmp)
*  01 apr 98 jen removed code to handle vldb - not needed
*            Upgraded version to 2.4         for ( chunk
*  01 apr 98 jen Fixed up import code on NT so the variable is recognized in
the
*            children
*  25 may 98 sks Reworked some of the environment variable code so
consolidate as
*            much as possible.  Not all complete because of differences in
*            the way nt and AIX calls (and starts stuff in background) for UFs
*  29 may 98 jen REUSE_STAGE Changed UF1 so we reuse the same staging
tables
*            instead of having a new set for each update pair
*  06 jul 98 jen Removed locking of staging tables since they are created with
*            locksize table now
*  06 jul 98 jen 912RETRY - added code to retry query execution on 912 as
well
*            as 911
*  07 jul 98 jen Fixed summary_table() so 1000x adjustment not based on UF
(setting
*            of max and min pointers
*            Added generic SleepSome function to handle NT vs AIX sleep
differences
*  01 apr 98 djd Added change to permit the use of table functions for UF1.
*            to enable this set TPCD_UPDATE_IMPORT to tf in TPCD.SETUP
file.
*            MERGED this into base copy on Jul 07
*  10 jul 98 jen haider's fix for 'outstream' var for error processing in
*            runUF1_fn and runUF2_fn
*            Updated version to 2.5
*  25 sep 98 jen Added stream number printing into mpqry* files and increases
*            accuracy of timestamp in mpqry (and mts*qry*) files
*  06 oct 98 jen TIME_ACC Added accuracy of timestamp in mpqry (and
mts*qry*)
*            files. Cleaned up misuse of Sleep and flushed buffers on
*            deadlocks
*  19 oct 98 kbs fix UF2_fn to correctly count rows deleted in case of deadlock
*  20 oct 98 kbs rewrite UF2 and UF2_fn for static SQL with staging table
*  23 oct 98 jen Cleaned up retrying of order/lineitem on lineitem deadlock in
UF1
*  24 oct 98 jen Used load_uf1 and load_uf2 instead of general load_updates
*  26 oct 98 kbs inject the UF1 with a single staging table
*  02 nov 98 jen Fixed processing of multiple chunks in uf2 so don't duplicate
*  21 nov 98 kmw Fixed BIGINT
*  05 dec 98 aph Moved runUF1_fn() and runUF2_fn() into a separate file
tpcdUF.sqc
*            so that it can be bound separately with a different isolation level.
*  21 dec 98 aph Integrated Jennifer's QppD calculation (rounding &
adjustment) fixes.
*  22 dec 98 aph For UFs during Throughput run, defer CONNECT until
children launched.
*  28 dec 98 aph Removed error_check() call after CONNECT RESET
*  29 dec 98 aph For UFs do not COMMIT in tpcdbatch.sqc.  COMMITs
happen in tpcdUF.sqc.
*  18 jan 99 kal replaced header with #include "tpcdbatch.h"
*  27 may 99 bbeaton from (03 mar 99 jen) Fixed SUN fix that wasn't
compatible with
*            NT (using %D %T instead of %x %X for strftime)
*  16 jun 99 jen Added missing LPCTSTR cast of semaphore file name for NT
*  17 jun 99 jen SEMA Changes semaphore file for update functions to look
*            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 tjg */
  unsigned int         stmt_num;                       /* @d24993 tjg */
```

```
  double          elapse_time;                    /* @d24993 tjg */
  double          adjusted_time;
  char            start_stamp[50];      /* start time stamp for block  */
  char            end_stamp[50];        /* end time stamp for block    */
  char            tag[50];              /* block tag                   */
  char            qry_description[100];
  struct stmt_info    *next;                      /* @d24993 tjg */

};


/*****************************************************************
********/
/* Structure containing command line options                    */
/*****************************************************************
********/
struct comm_line_opt
{                                       /* @d22275 tjg */
/* 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 tjg */
  int eo_infile;
  int time_stamp;
  int eo_block;                                 /* @d30369 tjg */
  int select_status;
};


/*****************************************************************
*********/
/* Function Prototypes                                          */
/*****************************************************************
*********/
int SleepSome( int amount );
int get_env_vars(void);
int Get_SQL_stmt(struct global_struct *g_struct);

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

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

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

void display_usage(void);
char *uppercase(char *string);
char *lowercase(char *string);
void comm_line_parse(int agrc, char *argv[], struct global_struct *g_struct);
int sqlrxd2a(char *decptr,char *asciiptr,short prec,short scal);
void init_setup(int argc, char *argv[], struct global_struct *g_struct);
void runUF1( 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_sqlda (struct sqlda *sqlda, int select_status);     /* @d30369 tjg */
void output_file(struct global_struct *g_struct);
int PreSQLprocess(struct global_struct *g_struct, Timer_struct *start_time);
void SQLprocess(struct global_struct *g_struct);
int PostSQLprocess(struct global_struct *g_struct, Timer_struct *start_time);
int cleanup(struct global_struct *g_struct);

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


EXEC SQL INCLUDE SQLCA;

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

char    stmt_str1[4000] = "\0";      /* Assume max SQL statment
                                        of 4000 char  */
struct {                              /* jen LONG */
       short len;
       char data[32700];
      } stmt_str;                     /* jen LONG */
char    dbname[9] = "\0";
char    userid[9] = "\0";
char    passwd[9] = "\0";
char    sourcefile[256];              /* used for semaphores and table functions?*/
sqlint32 chunk = 0;                   /* jenCI counter for within the set of chunks*/

EXEC SQL END DECLARE SECTION;

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

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

/* Other globals */
FILE            *instream, *outstream;  /* File pointers              */
int             verbose = 0;            /* Verbose option flag        */
int             semcontrol = 1;         /* allows/disallows smaphores usage */
```

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

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

/*****************************************************************
*****/
/* Start main program processing.                                */
/*****************************************************************
*****/
int main(int argc, char *argv[])
{
  /* kjd715 */
  /*struct comm_line_opt c_l_opt = { "\0","\0", 0, 1, 0, 0, 0 };*/ /* kjd715 */
  struct comm_line_opt c_l_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", "\0", NULL };
  /* first stmt_info structure */

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

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

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


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


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

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

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

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

  strcpy(temp_time_stamp, "0");
  /
*****************************************************************
***************
  *                                                               *
  *   This is the transition from the "driver" to the "SUT"                *
  *                                                               *
  *****************************************************************
****************/

  /
*****************************************************************
********/
/* Read in each statement, prepare, execute, and send output to file.  */
  /
*****************************************************************
********/

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

    c_flags.eo_block = 0;

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

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

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

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

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

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

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

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


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


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


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

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

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

 return(0);

}  /* end of main  */

/************************************************************
*********/
/* Generic form of Sleep */
int SleepSome( int amount)
{
#ifndef 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 tjg */
  int stmt_num_flag = 1;          /* first line of SQL stmt flag      */
  int eostmt = 0;               /* flag to signal end of statement  */


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

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

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

    if (strstr(input_ln,"--") == input_ln) {    /* Skip all -- comments */

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

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

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

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

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

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

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

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

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

      eostmt=0;
    }

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

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


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

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

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

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

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

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

  }  while (!eostmt);

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

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

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

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

  else                                            /* @d23684 tjg */
    return TPCDBATCH_NONSQL;

}  /*  Get_SQL_stmt */


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

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

  for (loopvar=0; loopvar<sqlda->sqld; loopvar++)
  {
    switch (sqlda->sqlvar[loopvar].sqltype)
    {
     case SQL_TYP_INTEGER:                        /* INTEGER */
     case SQL_TYP_NINTEGER:
       if ((sqlda->sqlvar[loopvar].sqldata=
              (TPCDBATCH_CHAR *)malloc(sizeof(sqlint32))) == NULL)
         mem_error("allocating INTEGER");
       break;
     case SQL_TYP_BIGINT:                         /* BIGINT */ /*kmwBIGINT*/
     case SQL_TYP_NBIGINT:
/*#ifdef SQLWINT */
/*       if ((sqlda->sqlvar[loopvar].sqldata= */
/*              (TPCDBATCH_CHAR *)malloc(sizeof(__int64))) == NULL)*/
/* #else */
       if ((sqlda->sqlvar[loopvar].sqldata=
              (TPCDBATCH_CHAR *)malloc(sizeof(sqlint64))) == NULL)
/* #endif*/
         mem_error("allocating BIGINT");
       break;
     case SQL_TYP_CHAR:                           /* CHAR */
     case SQL_TYP_NCHAR:
       if ((sqlda->sqlvar[loopvar].sqldata=
              (TPCDBATCH_CHAR *)calloc(256,sizeof(char))) == NULL)
         mem_error("allocating CHAR/VARCHAR");
       break;
     case SQL_TYP_VARCHAR:                        /* VARCHAR */
     case SQL_TYP_NVARCHAR:
       if ((sqlda->sqlvar[loopvar].sqldata=
              (TPCDBATCH_CHAR *)calloc(4002,sizeof(char))) == NULL)
         mem_error("allocating CHAR/VARCHAR");
       break;
```

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

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


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


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

  int    col_type;                /* Type of column            */

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

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


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

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

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

       case SQL_TYP_SMALL:
       case SQL_TYP_NSMALL:

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

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

          temp_ptr = temp_string;

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

          else
            strcpy(decimal_string, " ");

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

```
            ;

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

        break;

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

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

  fprintf(outstream, "\n");

  return;
}


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

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

#if defined (SQLOS2) || defined (SQLWINT)  || defined (SQLWIN) || defined
(SQLDOS)
  /*@d33143aha*/
  ftime (start_time);
#elif defined(SQLSNI)
  rc = gettimeofday(start_time);
#elif defined(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;
#ifndef SQLWINT
  long int           result_sec;
  long int           result_usec;
#endif


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

  }

  /*
   * translate the time to that rounded to the CLOSEST 0.1 seconds as
   * required by the TPC-D spec.   ROUNDING
   */
  /*   result = (double)(((long)((result + 0.099999) * 10))/10.0);*/
  result = (double)(((long)((result + 0.05) * 10))/10.0);
  return (result);
}


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

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


/*************************************************************
***************/
/* error_check                                          */
/* This function prints the contents of the sqlca error information          */
/* structure.                                           */
/*************************************************************
***************/
long error_check(void)
```

```
{
  char          buffer[512]="\0";
  unsigned short i;
  struct sqlca   temp_sqlca;       /* temporary sqlca */     /* @d30369 tjg */

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

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

    /* Decode the SQLCA in more detail  KBS 98/09/28 */
    if ((sqlca.sqlerrml)   /*  there's one or more tokens   */
       && (sqlca.sqlerrml < sizeof(sqlca.sqlerrmc)) /* and field not full */
       )
    {
      char *tokptr;
      int  tokl;
      *(sqlca.sqlerrmc + sqlca.sqlerrml) = '\0';  /*  prevent strtok from
scanning beyond end */
      fprintf(stderr,"\n     SQLCA:   tokens:\n");
      fprintf(outstream,"\n     SQLCA:   tokens:\n");
      tokptr=strtok(sqlca.sqlerrmc, "\xff");
      while (  tokptr                          &&
            ( (tokl = (sizeof(sqlca.sqlerrmc) - (tokptr-sqlca.sqlerrmc))) > 0)
            )
      {
        fprintf(stderr, "%.*s\n", tokl, tokptr);
        fprintf(outstream, "%.*s\n", tokl, tokptr);
        tokptr=strtok(NULL, "\xff");
      }
    }
    fprintf(stderr, "\n     SQLCA:   errp= %.8s, errd 1-6= %d %d %d %d %d %
d\n",
          sqlca.sqlerrp, sqlca.sqlerrd[0], sqlca.sqlerrd[1], sqlca.sqlerrd[2],
          sqlca.sqlerrd[3], sqlca.sqlerrd[4], sqlca.sqlerrd[5]);
    fprintf(outstream, "\n     SQLCA:   errp= %.8s, errd 1-6= %d %d %d %d %
d %d\n",
          sqlca.sqlerrp, sqlca.sqlerrd[0], sqlca.sqlerrd[1], sqlca.sqlerrd[2],
          sqlca.sqlerrd[3], sqlca.sqlerrd[4], sqlca.sqlerrd[5]);

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

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

    EXEC SQL CONNECT ;                          /* @d28763 tjg */

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

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

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

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

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

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

    }
  }
  return (temp_sqlca.sqlcode);

}  /* error_check */
```

```
/*****************************************************/
/* Displays a help screen                            */
/*****************************************************/
void display_usage()
{
  printf("\ntpcdbatch  --  version %s",TPCDBATCH_VERSION);
  printf("\n\nSyntax is:\n");
  printf("tpcdbatch [-d dbname] [-f file_name] [-l file_name] [-r on/off]");
  printf("\n        [-v on/off] [-b on/off] [-u p/t/t1/t2]");
  printf("\n        [-s scale_factor] [-n stream_num] [-m inlistmax] [-h]\n");
  printf("\n  where: -d  Database name");
  printf("\n                Default - dbname set in $DB2DBDFT");
  printf("\n        -f  Input file containing SQL statements");
  printf("\n                Default - stdin ");
  printf("\n        -r  Create set of output files containing query results");
  printf("\n                Default - off");
  printf("\n        -v  Verbose.  Sends information to stderr during");
  printf("\n            query processing");
  printf("\n                Default - off");
  printf("\n        -b  Process groups of statements as blocks ");
  printf("\n            instead of individually.");
  printf("\n                Default - off");
  printf("\n        -u  Update streams: p    - for power test");
  printf("\n                            t    - for throughput test without");
  printf("\n                                   UFs (run this instead of t2)");
  printf("\n                            t1   - for throughput test step 1");
  printf("\n                                   only running queries");
  printf("\n                            t2   - for throughput test step 2");
  printf("\n                                   running update functions");
  printf("\n        -s  Scale factor");
  printf("\n                Default - 0.1");
  printf("\n        -n  Stream number");
  printf("\n                      Default - 0");
  printf("\n                Qualification - -1");
  printf("\n                        Power - 0");
  printf("\n                  Throughput - >= 1 (actual number depends on the
current query stream");
  printf("\n        -m  Maximum number of keys to delete at a time");
  printf("\n                Default - 400");
  printf("\n        -h  Display this help screen");
  printf("\n        -p  turns smeaphores on or off");
  printf("\n                Default - off");

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

  printf("\n\nSyntax:    --#SET <control option> <value>");
  printf("\n\n  option      value    default");
  printf("\nROWS_FETCH   -1 to n     -1  (all rows fetched from answer set)");
  printf("\nROWS_OUT     -1 to n     -1  (all fetched rows sent to output)");
  printf("\n\n--#TAG       tag            (user specified tag name for sequence#)");
  printf("\n--#COMMENT   comment         (user specified comments for
output)");
  printf("\nNote:  All statements executed with ISOLATION LEVEL RR");
  printf("\n       and must be terminated with semi-colons.\n");
  exit (1);
}


/*****************************************************/
/* Converts a string to upper case characters   */
/*****************************************************/
char *uppercase( char *string )
{
  char   *c;     /* temp char used to convert word to upper case */

  for ( c = string; *c != '\0'; c++)
    *c = (char) toupper( (int) *c );

  return (string);
}

/*****************************************************/
/* Converts a string to lower case characters   */
/*****************************************************/
char *lowercase( char *string )
{
  char   *c;     /* temp char used to convert word to lower case */

  for ( c = string; *c != '\0'; c++)
    *c = (char) tolower( (int) *c );
```

```
  return (string);
}


/*****************************************************/
/* Parses and processes command line options.      */
/*****************************************************/

void comm_line_parse(int argc, char *argv[], struct global_struct *g_struct)
{
  char authent_info[40] = "\0";
  char *testptr;
  int loopvar = 0;

  int comm_opt = 0;
#ifdef PARALLEL_UPDATES
  int running_updates=0;
  int updatePair=-1;
  int updateStream=-1;
  int function;
  int copyOnOrOff;
  int deleteChunk=0;        /*DELjen */
#endif

  while ((loopvar < argc) && (argc != 1)) {

    if (*argv[loopvar] == '-') {

      switch(*(argv[loopvar]+1)) {

       case 'f' :                          /* @d26350 tjg */
       case 'F' :
                 strcpy(g_struct->c_l_opt->infile,argv[++loopvar]);
                 break;
       /* kjd715 */
       case 'l' :
       case 'L' :   loopvar+=1;
                                  /*
                                  strcpy(g_struct->c_l_opt->str_file_name,argv
[++loopvar]);
                                  */
                 break;
            /* kjd715 */
       case 'r' :                          /* @d26350 tjg */
       case 'R' :
         if (!strcmp(uppercase(argv[++loopvar]),"ON"))
           g_struct->c_l_opt->outfile=1;
         else
           g_struct->c_l_opt->outfile=0;
         break;

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

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

       case 'u' :                          /* @d26350 tjg */
       case 'U' :
         g_struct->c_l_opt->update=-1; /* init to invalid number */
         if (!strcmp(uppercase(argv[++loopvar]),"P1"))
           g_struct->c_l_opt->update=1; /* power query stream*/
         if (!strcmp(uppercase(argv[loopvar]),"P2"))
              g_struct->c_l_opt->update=3; /* power update with updates*/
         if (!strcmp(uppercase(argv[loopvar]),"P"))
           g_struct->c_l_opt->update=4; /* power update without updates*/
         if (!strcmp(uppercase(argv[loopvar]),"T1"))
              g_struct->c_l_opt->update=0; /*throughput query stream */
         if (!strcmp(uppercase(argv[loopvar]),"T2"))
              g_struct->c_l_opt->update=2; /* throughput update with updates */
         if (!strcmp(uppercase(argv[loopvar]),"T"))
              g_struct->c_l_opt->update=5; /* throughput update without
updates */
         break;

       case 'b' :                          /* @d26350 tjg */
       case 'B' :
         if (!strcmp(uppercase(argv[++loopvar]),"ON"))
           g_struct->s_info_ptr->query_block=1;
         else
           g_struct->s_info_ptr->query_block=0;
         break;

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

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

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

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

       case 'p' :
       case 'P' :
          if (!strcmp(uppercase(argv[++loopvar]),"ON")) /* bbe 072599 */
            semcontrol = 1;
          else
            semcontrol = 0;
          break;


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

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

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

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

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

      }  /** end switch **/
    } /** end if **/
```

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

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


#ifdef PARALLEL_UPDATES
  if (running_updates) {
    if (updatePair == -1) {
      fprintf (stderr, "The parameters to tpcdbatch have not been passed
correctly\n");
      exit (-1);
    }
    else {
      /* check to see if we are to use copy on for the load */
      if (( getenv("TPCD_LOG") != NULL ) &&
         (!strcmp(uppercase(getenv("TPCD_LOG")),"YES")))
      {
        /* okay, we have set LOG_RETAIN on so we need to use copy
directory */
        copyOnOrOff = TRUE;
      }
      else
      {
        /* log retain off don't use copy directory */
        copyOnOrOff = FALSE;
      }

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

  /* If no database name is given, then use the one specified in the
     environment variable DB2DBDFT, otherwise error */
  if (!strcmp(dbname,"\0")) {
    testptr = getenv("DB2DBDFT");
    if (testptr == NULL) {
      fprintf(stderr, "\nNo database name has been specified on command ");
      fprintf(stderr, "line\nnor in environment variable DB2DBDFT.");
      display_usage();
    }
    else
      strcpy(dbname,testptr);

}
/* 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,"\0")) {
    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%smseedme", env_tpcd_audit_dir,
          env_tpcd_path_delim, env_tpcd_path_delim);

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

  /* check to see if we are to use copy on for the load */
  if (( getenv("TPCD_LOG") != NULL ) &&
     (!strcmp(uppercase(getenv("TPCD_LOG")),"YES")))
  {
    /* okay, we have set LOG_RETAIN on so we need to use copy directory */
    g_struct->copy_on_load = TRUE;
  }
  else
  {
    /* log retain off don't use copy directory */
    g_struct->copy_on_load = FALSE;
  }

/*************************************************************
***/
/* 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");
    sqlestar ();
  }
  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");
        sqlestar ();
        connect=0;
      }
      else connect=1;
    } while (!connect);
```

```
    error_check();
  }


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

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

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

  /* if we are running a throughput test, then we must report the */
  /* stream count information...we will report one file per stream */
  /* and amalgamate them after all streams have completed */
  /* if the number of streams is greater than 0 then this is a throughput test*/
  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%spstrcntuf.%s",g_struct->run_dir,
                  env_tpcd_path_delim, g_struct->file_time_stamp);
            break;
      case (1):
            /* power query stream */
            sprintf(file_name, "%s%spstrcnt%d.%s",g_struct->run_dir,
env_tpcd_path_delim,
                  g_struct->c_l_opt->intStreamNum,g_struct->file_time_stamp);
            break;
      case (0):
            /* throughput query stream */
            sprintf(file_name, "%s%sstrcnt%d.%s",g_struct->run_dir,
env_tpcd_path_delim,
                  g_struct->c_l_opt->intStreamNum,g_struct->file_time_stamp);
            break;
  }

  if( (g_struct->stream_report_file = fopen(file_name, WRITEMODE)) ==
NULL )
  {
    fprintf(stderr,"\nThe output file for the stream count information\n");
    fprintf(stderr,"could not be opened, make sure the filename is correct\n");
    fprintf(stderr,"filename = %s\n",file_name);
    exit(-1);
  }

  if (g_struct->c_l_opt->update > 1)
  {
    /* update function stream */
    fprintf(g_struct->stream_report_file,
          "Update function stream starting at %*.*s\n",
          T_STAMP_3LEN,T_STAMP_3LEN, /* TIME_ACC jen*/
          get_time_stamp(T_STAMP_FORM_3,&(g_struct-
>stream_start_time))); /* TIME_ACC jen*/
  }
  else
  {
    /* query stream */
    fprintf(g_struct->stream_report_file,
          "Stream number %d starting at %*.*s\n",
          g_struct->c_l_opt->intStreamNum,
          T_STAMP_3LEN,T_STAMP_3LEN,          /* TIME_ACC jen*/
          get_time_stamp(T_STAMP_FORM_3,&(g_struct-
>stream_start_time))); /* TIME_ACC jen*/
  }

#ifndef LINUX

  fclose(g_struct->stream_report_file);

#endif

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

  sprintf(g_struct->update_num_file, "%s%s%s.%s.update.pair.num",
env_tpcd_audit_dir,
        env_tpcd_path_delim, uppercase(env_tpcd_dbname), lowercase
(env_user));
  sprintf(g_struct->sem_file, "%s.%s.semfile", env_tpcd_dbname, env_user);
  if (g_struct->c_l_opt->intStreamNum == 0)
  {
    sprintf(g_struct->sem_file2, "%s.%s.semfile2", env_tpcd_dbname,
env_user);
  }
  if (verbose) {   /* print out the update pair number file for debugging */
    fprintf(stderr,"\n init_setup: strem %d update pair numb file = %s\n",
          g_struct->c_l_opt->intStreamNum,g_struct->update_num_file);
  }

  /* update the
$TPCD_AUDIT_DIR/$TPCD_DBNAME.$USER.update.pair.num file */
  /* update pairs have been run */
  if (( g_struct->c_l_opt->update >= 1 ) && ( g_struct->c_l_opt->update <
4 ))
      /* on or onl, but not  */ /* bbe or > 1 */
  {
    updateFP = fopen(g_struct->update_num_file,"r");
    if (updateFP != NULL )
    {
      fscanf(updateFP,"%d",&updatePairStart);
      fclose(updateFP);
      if (g_struct->c_l_opt->intStreamNum == 0)   /* on, 1 update pair */
        updatePairStop = updatePairStart + 1;
      else       /* only, multiple update pairs, stream number will be total */
        updatePairStop = updatePairStart + g_struct->c_l_opt->intStreamNum;
      currentUpdatePair = updatePairStart;

      if (updatePairStart <= 0)
      {
        fprintf(stderr,"updatePairStart is bogus!");
        exit(-1);
      }
    }
    else
    {
      fprintf(stderr,"\n %s not set up, set this \n",g_struct->update_num_file);
      fprintf(stderr,"file to contain the number of the update pair to  \n");
      fprintf(stderr,"run and resubmit\n");
      exit(-1);
    }
  }

  return ;

}

/*****************************************************************
*******/
/* A function to print out the column titles for a returned set      */
/*****************************************************************
*******/
void print_headings (struct sqlda *sqlda, int *col_lengths)
{
  int col = 0;                    /* Column number            */
  int col_width = 0;              /* width of column          */
  int max_col_width = 0;          /* maximum column width     */
  int col_name_length = 0;        /* sizeof column name string */
  int col_type = 0;               /* column type              */

  int total_length = 0;           /* accumulator var. for
                                  length of column headings */
  int loopvar = 0;

  char col_name[256] = "\0";
  unsigned char m,n;              /* precision and accuracy
                                  for decimal conversion    */
```

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

  /** loop through for each column in solution set
    and determine the maximum column width  **/

  for (col = 0; col < sqlda->sqld; col++) {
    col_name_length=sqlda->sqlvar[col].sqlname.length;
    col_type = sqlda->sqlvar[col].sqltype;
    col_width = sqlda->sqlvar[col].sqllen;
    strncpy(col_name,(char *)sqlda->sqlvar[col].
sqlname.data,col_name_length) ;

    switch (col_type)
    {
     case SQL_TYP_SMALL:
     case SQL_TYP_NSMALL:                          /* @d30369 tjg */
       col_lengths[col] = TPCDBATCH_MAX (col_name_length,6);
       break;
     case SQL_TYP_INTEGER:
     case SQL_TYP_NINTEGER:
       col_lengths[col] = TPCDBATCH_MAX (col_name_length,11);
       break;
     case SQL_TYP_BIGINT:   /*kmwBIGINT*/
     case SQL_TYP_NBIGINT:
       col_lengths[col] = TPCDBATCH_MAX (col_name_length,19);
       break;
     case SQL_TYP_CSTR:
     case SQL_TYP_NCSTR:
     case SQL_TYP_DATE:
     case SQL_TYP_NDATE:
     case SQL_TYP_TIME:
     case SQL_TYP_NTIME:
     case SQL_TYP_STAMP:
     case SQL_TYP_NSTAMP:
     case SQL_TYP_CHAR:
     case SQL_TYP_NCHAR:
     case SQL_TYP_VARCHAR:
     case SQL_TYP_NVARCHAR:
     case SQL_TYP_LONG:
     case SQL_TYP_NLONG:
       col_lengths[col] = TPCDBATCH_MAX (col_name_length,col_width);
       break;

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

     case SQL_TYP_DECIMAL:
     case SQL_TYP_NDECIMAL:

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

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

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

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

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

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

}


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

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

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

  if ((form == T_STAMP_FORM_1) || (form == T_STAMP_FORM_3))
  {
  /* SUN fix bbe start */
#if (defined (SQLWINT) || defined (SQLWIN) || defined (SQLOS2) || defined
(SQLDOS))
    timeLength = strftime(newtime,50,"%x %X",tp);
#elif (defined (SQLUNIX) || defined (SQLAIX))
    timeLength = strftime(newtime,50,"%D %T",tp);  /* SUN  ...test this */
#else
#error Unknown operating system
#endif
  /* SUN fix bbe stop */
    /* TIME_ACC jen start*/
    if (form == T_STAMP_FORM_3)
    {
      /* concatenate the microsecond/milliseconds  on the end of the */
      /*timestamp jen1006 */
#if defined (SQLUNIX) || defined (SQLAIX)
      sprintf(newtime+timeLength,".%0.6d",temp_stamp.tv_usec);
#elif (defined (SQLOS2) || defined(SQLWINT) || defined (SQLWIN) || defined
(SQLDOS))
      sprintf(newtime+timeLength,".%0.3d",temp_stamp.millitm);
#else
#error Unknown operating system
#endif
      /* TIME_ACC jen stop*/
    }
  }
  else
    if (form == T_STAMP_FORM_2)
      strftime(newtime,50,"%y%m%d-%H%M%S",tp);

  return (newtime);

}


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

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

  double adjusted_a_mean = 0;
  double adjusted_g_mean = 0;
  double adjusted_g_mean_intern;
  double adjusted_max_time = 0;

  double Ts   = 0;                /* different TPC-D metrics */
  double Ts1;
  double Ts2;
```

```
/* double QppD = 0;                    MARK
   double QthD = 0;
   double QphD = 0; */

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

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


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


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

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

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

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

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

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

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

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

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

     s_info_ptr = s_info_ptr->next;

   }

   s_info_ptr = s_info_head_ptr;

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

     if (s_info_ptr->elapse_time != -1)  {
       s_info_ptr->adjusted_time = s_info_ptr->elapse_time;
       /* determine whether the elapsed times have to be adjusted or not */
       /* if this is an update function, we do not adjust the elapsed time*/
       if ( strstr(s_info_ptr->tag,"UF") == NULL )
       {
         /* this is not an update function, adjust time if necessary */
         if (max->elapse_time/min->elapse_time > 1000)
         {
           /* jmc fix geo_mean calculation...round adjusted time properly
ROUNDING*/
           adjusted_max_time = max->elapse_time/1000;
           if (s_info_ptr->elapse_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->elapse_time,s_info_ptr->adjusted_time,
             T_STAMP_1LEN,T_STAMP_1LEN,s_info_ptr->start_stamp, /*
TIME_ACC jen*/
             T_STAMP_1LEN,T_STAMP_1LEN,s_info_ptr->end_stamp); /*
TIME_ACC jen*/

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

       if (s_info_ptr->elapse_time > 0) {  /* don't bother finding log of
                                              numbers < 0 */
         geo_mean += log(s_info_ptr->elapse_time);
         adjusted_g_mean += log(s_info_ptr->adjusted_time);
       }


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

```
      num_stmt_for_geo_mean++;
    }

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


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

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

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

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

#if defined (SQLUNIX) || defined (SQLAIX)
    Ts = g_struct->stream_end_time.tv_sec - g_struct-
>stream_start_time.tv_sec + 1;
    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 calcmetricsp.pl##
    QppD = (3600 * g_struct->scale_factor) / adjusted_g_mean;
    QthD = (num_stmt * 3600 * g_struct->scale_factor) / Ts;
    QphD = sqrt(QppD*QthD);
   */
    /* if the decimal part has some meaningful value then print the database
size
    with decimal part; otherwise just print the integer part */

      fprintf (outstream,
           "\nGeometric mean interim value  = %10.3f\n\nStream Ts %11 = %
10.0f\n\nStream start int representation %11 = %f\n\nStream stop int
representation  %11 = %f",
           adjusted_g_mean_intern,Ts,Ts1,Ts2);
  }

}


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

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

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


/**********************************************/
/* processing to run the insert update function */
/**********************************************/
void runUF1 ( 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_semfile[256];
#else
  int childpid[100];
  int               su_semid; /* semaphore for controlling split updates*/
  key_t             su_semkey; /* key to generate semid */
#endif
```

```
  if (g_struct->c_l_opt->intStreamNum == 0)
     streamNum = 0;
  else
     streamNum = currentUpdatePair - updatePairStart + 1;

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

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

  /* Load the orderkeys into the staging table */
  /* In SMP environments one could use a load command but by using a */
  /* script we can keep the code common */
#ifdef SQLWINT
  sprintf (statement, "perl %s\\tools\\ploaduf1 %d\n", sourcedir, updatePair);
#else
  sprintf (statement, "perl %s/tools/ploaduf1 %d 1", sourcedir, updatePair);
#endif
  if (system(statement))
  {
    fprintf (stderr, "ploaduf1 failed for UF1, examine UF1.log for cause.
Exiting.\n");
     if (verbose)
       fprintf (stderr,
           "ploaduf1 failed for UF1, examine UF1.log for cause. Exiting.\n");
    exit (-1);
  }

  fprintf (outstream, "load_update finished for UF1.\n");

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

#ifndef 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*/
  {
#ifndef SQLWINT
    if ((childpid[i] = fork()) == 0)
    {
      /* runUF1_fn (updatePair, i);   aph 981205 */
      runUF1_fn (updatePair, i, dbname, userid, passwd);
    }
    else
    {
      /* This is the parent */
      if (verbose)
        fprintf (stderr, "stream #%d started with pid %d\n", i, childpid[i]);
    }
#else  /* SQLWINT */
    sprintf (commandline,
          "start /b %s\\auditruns\\tpcdbatch.exe -z -d %s -i %d -j 1 -k %d",
          env_tpcd_audit_dir, dbname, updatePair, i ); /* aph 082797 */

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

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

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

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


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


/*************************************************/
/* processing to run the delete update function */
/*************************************************/
void runUF2 ( 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 */
#ifndef 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++)
  {
#ifndef SQLWINT
    if ((childpid[i] = fork()) == 0)
    {
      fprintf(stderr, "B-Calling runUF2_fn  %d   %d   %d ...\n",
                    updatePair, i,chunks_per_concurrent_delete);
      /* runUF2_fn (updatePair, i, chunks_per_concurrent_delete);   aph
981205 */
      runUF2_fn (updatePair, i, chunks_per_concurrent_delete, dbname,
userid, passwd);
    }
    else
    {
      /* This is the parent */
      if (verbose)
        fprintf (stderr, "stream #%d started with pid %d\n", i, childpid[i]);
    }
#else
    {
      /*  SECURITY_ATTRIBUTES sec_process;
          SECURITY_ATTRIBUTES sec_thread; */
      /* NEED TO FIX THIS UP - KBS 98/10/20 */

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

  /* All children have been created, now wait for them to finish */
#ifndef SQLWINT
  fprintf(stderr, "About to wait on the semaphore...\n");
  if (sem_op (su_semid, 0, concurrent_deletes * -1) != 0)              /
*jenSEM*/
  {                                                          /*jenSEM*/
    fprintf(stderr,
          "Failure to update wait on delete semaphone with %d children\n",
          concurrent_deletes);
```

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

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

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

}


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


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

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

/*****************************************************************
****/
/* Determines the proper name for the output file to
  be generated for a particular TPC-D query, update function, or
  interval summary                                          */
/*****************************************************************
****/
void output_file(struct global_struct *g_struct)
{
  char file_name[256] = "\0";
  char run_dir[150]   = "\0";
  char time_stamp[50] = "\0";
  char delim[2]       = "\0";
  int qnum=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_l_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: /* qualifiying */
     sprintf(file_name, "%s%sqryqual%02d.%
s",run_dir,delim,qnum,time_stamp);
     break;
   case 0: /* power tests */
     if (qnum < 0) /* update functions */
       sprintf(file_name, "%s%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;
#ifndef 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*/
#ifndef NO_INCREMENT
        /* don't update the pair, only for my testing - Haider */
        updateFP = fopen(g_struct->update_num_file,"w");
        fprintf(updateFP,"%d\n",currentUpdatePair);
        fclose(updateFP);
#endif
    }  /*jen*/
    rc = FALSE;
    break;

  }
  return(rc);
}


/*****************************************************************
*******/
/* Handles actual processing of SQL statement.  Initializes the SQLDA
  for returned rows, does PREPARE, DECLARE, and OPEN statements and
  executed multiple FETCHes as needed.  If not a SELECT statement,
  goes into EXECUTE IMMEDIATE section                          */
/*****************************************************************
*******/
void SQLprocess(struct global_struct *g_struct)
{
  int rc = 0;                        /* 912RETRY */
  int rows_fetch = 0;
  long sqlcode = SQL_RC_E911;             /* Temporary sqlcode to test
                                    for deadlocks */
  int max_wait = 1;                       /* Maximum number of retries
                                    for deadlock scenario */

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

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

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


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

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

    sqlcode = 0;          /* Re-initialize sqlcode to avoid infinite-loop */
    if (g_struct->c_flags->select_status == TPCDBATCH_SELECT)
    {
      /* Enter this loop if SQL stmt is a SELECT   */
      EXEC SQL PREPARE STMT1 INTO :*sqlda FROM :stmt_str;
      sqlcode = error_check();
      if (sqlcode < 0)
      {
        fprintf (stderr,"\nPrepare failed. Stopping this query.\n");
        rc = -1;
      }
      else  /* print out the column headings for the answer set */
      {
        print_headings(sqlda,col_lengths);              /* @d22817 tjg */

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

        EXEC SQL DECLARE DYNCUR CURSOR FOR STMT1;

        EXEC SQL OPEN DYNCUR;
        sqlcode = error_check();

        if (sqlcode < 0)      /* we ran into an error of some kind KBS 98/09/28
*/
        {
          max_wait ++;
          fprintf (stderr, "\nAn error has been detected on
open...Retrying...\n");
          SleepSome(10);
        }
        else
        {
          /
*****************************************************************
********/
          /* Fetch appropriate number of rows and determine whether or not to
*/
          /* send them to file.                                  */
          /
*****************************************************************
********/

          rows_fetch = 0;

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

            EXEC SQL FETCH DYNCUR USING DESCRIPTOR :*sqlda;
            if (sqlca.sqlcode == 100)
            {
              sqlcode = sqlca.sqlcode;
            }
            else
            {
              sqlcode = error_check();
            }
            if (sqlcode == 0)
            {
              rows_fetch++;
              if ( (rows_fetch <= s_info_ptr->max_rows_out) ||
                  (s_info_ptr->max_rows_out == -1) )
                echo_sqlda(sqlda,col_lengths);
            }
            else if (sqlcode < 0)
            {
              max_wait++;
              fprintf (stderr, "\nAn error has been detected on
fetch...Retrying...\n");
              SleepSome(10);
            }
          }  while ( (sqlcode == 0) && \
                  ( (s_info_ptr->max_rows_fetch == -1) || \
                    (rows_fetch < s_info_ptr->max_rows_fetch) ) );
        } /* end of successful open */
      } /* end of successful prepare */
    } /** End of block for handling SELECT statements **/

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

```
      if ((sqlcode < 0 ) && (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))
#ifndef 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 tjg */

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

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

} /* end of SQLprocess */

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

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

  s_info_ptr = g_struct->s_info_ptr;


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

  if (g_struct->c_l_opt->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 tjg */
   }
  }

  fflush(outstream);

  s_info_ptr->elapse_time = get_elapsed_time(start_time);

  if (g_struct->c_flags->time_stamp == TRUE)          /* @d25594 tjg */
  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->elapse_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->elapse_time = -1;

  s_info_ptr = s_info_ptr->next;

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

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

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

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

return (TRUE);
}

/*****************************************************************
****************/
/* Does some cleaning up once all the statements are processed.  Disconnects
  from the database, cleans up some semaphore stuff from the update
functions,
  prints out the summary table, and closes all file handles.              */
/*****************************************************************
****************/
int cleanup(struct global_struct *g_struct)
{
#ifndef SQLWINT
  int              semid;            /* semaphore for controlling UFs*/
  key_t            semkey;           /* key to generate semid */
#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%spstrcntuf.%s",g_struct->run_dir,
                  env_tpcd_path_delim, g_struct->file_time_stamp);
            break;
      case (1):
            /* power query stream */
            sprintf(file_name, "%s%spstrcnt%d.%s",g_struct->run_dir,
env_tpcd_path_delim,
                  g_struct->c_l_opt->intStreamNum,g_struct->file_time_stamp);
            break;
      case (0):
            /* throughput query stream */
            sprintf(file_name, "%s%sstrcnt%d.%s",g_struct->run_dir,
env_tpcd_path_delim,
                  g_struct->c_l_opt->intStreamNum,g_struct->file_time_stamp);
            break;
   }

#ifndef 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 */
#ifndef SQLWINT
     if (verbose)
     {
      fprintf(stderr,
            "cleanup: semkey = %ld, semid = %d, file = %s, stream = %d\n",
            semkey,semid,g_struct->update_num_file,
            g_struct->c_l_opt->intStreamNum);
     }
#endif
   }


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

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

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

   return (TRUE);
}

void create_semaphores(struct global_struct *g_struct)
{

#ifndef 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);
                                        exit(1);
                                }
                        }
                        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)
{
#ifndef SQLWINT
  int         semid;          /* semaphore for controlling UFs*/
  key_t       semkey;         /* key to generate semid */
#else
  HANDLE      hSem;
  int         j;
  int         SemTimeout = 600000;    /* Des time out period of 1 minute */
#endif

#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 semaphone 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)
{
#ifndef SQLWINT
   int              semid;            /* semaphore for controlling UFs*/
   key_t            semkey;           /* key to generate semid */
#else
   HANDLE           hSem;
   int              SemTimeout = 600000;   /* Des time out period of 1 minute
*/
#endif

#ifdef SQLWINT
      hSem = open_semaphore(g_struct, sem_num); /* query */
      if (! ReleaseSemaphore(hSem,
                        1,
                        (LPLONG)(NULL)))
      {
         fprintf(stderr, "ReleaseSemaphore failed, Sem#: %d LastError: %d,
quit\n",
                  sem_num, GetLastError());
            exit(-1);
      }
#else
      semid = open_semaphore(g_struct); /* query */
      /* aix semaphores start at 0, not 1, so sem_num -1 is used */
      if (sem_op(semid, sem_num - 1, 1) != 0)                    /*jenSEM*/
       {                                        /*jenSEM*/
            fprintf(stderr,
                  "Failed to increment semaphore %d for throughput stream %
d\n",
                  sem_num, g_struct->c_l_opt->intStreamNum);
            fprintf(stderr,
                  "file for generation of semaphore is: %s\n",
                  g_struct->update_num_file);
            exit(1);
       }

#endif
      if (g_struct->c_l_opt->intStreamNum == 0)
      {  /* RUNPOWER */
        if (sem_num == 1)
        {
          fprintf(stderr, "UF1 completed.\n");
        }
        else
        {
          fprintf(stderr, "query stream completed.\n");
        }
      }
}

#ifdef SQLWINT /* Compile only in NT */
HANDLE open_semaphore(struct global_struct *g_struct, int num)
{
      HANDLE  hSem;
      LPCTSTR semfile;

      if (num == 1)
            semfile = (LPCTSTR)g_struct->sem_file;
      else
            semfile = (LPCTSTR)g_struct->sem_file2;

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

        Sleep(1000);
      }
      return hSem;
}

#else  /* Compile only in non-NT (i.e. AIX) */
int open_semaphore(struct global_struct *g_struct)
{
      int              semid;            /* semaphore for controlling UFs*/
      key_t            semkey;           /* key to generate semid */
      int num;

      if (g_struct->c_l_opt->intStreamNum == 0)
            num = 2;
      else
            num = 1;

      semkey = ftok(g_struct->update_num_file,'J');
      while ((semid = semget(semkey,num,0)) < 0)
      {
            if (errno == ENOENT)
            {
                  sleep(2);
                  fprintf(stderr,"cleanUp: looping for access to semaphore stream
%d  ",
                        g_struct->c_l_opt->intStreamNum);
                  fprintf(stderr,"semkey=%ld semid = %d file=%
```

```
s\n",semkey,semid,
                    g_struct->update_num_file);
            }
            else
            {
                fprintf(stderr,"query stream %d semget failed errno = %d\n",
                    g_struct->c_l_opt->intStreamNum,errno);
                exit(1);
            }
        }
        return semid;
}
#endif
#
```

getvars

```
#!/usr/bin/perl

##################################
# Script to read the tpcd.setup and tpcd.runsetup
# file to get the environment variables
##################################

# open the file that has the "one time" setup variables
open (SETTINGS,"tpcd.setup") ||
        die "Cannot open the audit setting file tpcd.setup, please create";

&parseVars(SETTINGS);
close (SETTINGS);

@reqVars    =  ("TPCD_PLATFORM",
                "TPCD_PRODUCT",
                "TPCD_VERSION",
                "TPCD_DBNAME",
                "TPCD_MODE",
                "TPCD_SF",
                "TPCD_DDLPATH",
                "TPCD_AUDIT",
                "TPCD_AUDIT_DIR");

#print ("before setVar call\n");
&setVar(@reqVars, "ERROR");
#print ("after setVar call\n");


$version=  "tpc-h/tpc-r package 2.9 February 23 2001";
$platform=          $ENV{"TPCD_PLATFORM"};
$product= $ENV{"TPCD_PRODUCT"};
$multinode=         $ENV{"TPCD_RUN_ON_MULTIPLE_NODES"};
$defSeed=$ENV{"TPCD_GENERATE_SEED_FILE"};
$dbname= $ENV{"TPCD_DBNAME"};
$qualdbname=        $ENV{"TPCD_QUAL_DBNAME"};
$tpcdVersion=       $ENV{"TPCD_VERSION"};

# translate to lower case
$platform =~ tr/A-Z/a-z/;
$product =~ tr/A-Z/a-z/;
$multinode =~ tr/A-Z/a-z/;
$defSeed =~ tr/A-Z/a-z/;

# translate to upper case
$dbname =~ tr/a-z/A-Z/;
$qualdbname =~ tr/a-z/A-Z/;

# determine the number of queries
if ( $tpcdVersion == 1 ){
   $numQueries = 17;
   $ordTblName = "order";
   $qualSF=0.1;
}
elsif ( $tpcdVersion == 2){
   $numQueries = 22;
   $ordTblName = "orders";
   $qualSF = 1.0;
}
else{
   # set defaults
   $tpcdVersion=1;
   $numQueries=17;
   $ordTblName = "order";
   $qualSF = 0.1;
}

#determine the delimiter
if ( $platform eq "nt" ){
   $WINT=1;
   $delim="\\";
   $user=$ENV{"USERNAME"};
   $ENV{"USER"} = $user;
   $sep = "&";
}
elsif ( $platform eq "os2" ){
   $OS2=1;
   $delim="\\";
   $sep = ";";
}
elsif ( $platform eq "aix" || $platform eq "linux" || $platform eq "sun" ||
$platform eq "ptx" || $platform eq "hp"){
  $AIX=1;
  $delim="/";
  chop($OS=`uname`);
  umask 002;
  $user=$ENV{"USER"};
  $sep = ";";
}
else{
  die "ERROR: platform '$platform' not supported\n";
}

# make it viewable outside
$ENV{"TPCD_PACKAGE_VERSION"} =                    $version;
$ENV{"TPCD_VERSION"} =                    $tpcdVersion;
$ENV{"TPCD_PATH_DELIM"} =                 $delim;
$ENV{"TPCD_PLATFORM"} =                   $platform;
$ENV{"TPCD_PRODUCT"} =                            $product;
$ENV{"TPCD_RUN_ON_MULTIPLE_NODES"} =      $multinode;
$ENV{"TPCD_GENERATE_SEED_FILE"} =         $defSeed;
$ENV{"TPCD_QUAL_DBNAME"} =                $qualdbname;
$ENV{"TPCD_DBNAME"} =                     $dbname;
$ENV{"TPCD_NUM_QUERIES"} =                $numQueries;
$ENV{"TPCD_ORDTBL_NAME"} =                $ordTblName;
$ENV{"TPCD_QUAL_SF"} =                    $qualSF;
$ENV{"DB2OPTIONS"} =                      "+c -t -v";
$ENV{"COMMAND_SEP"} =                             $sep;

# open the file that has the "run specific" setup variables.  NOTE: this file
# will not exist if setupRun has not been called.  This is not an error, but
# a warning message will be sent that says setupRun has not been called yet.

#print ("before open settings tpcd.runsetup\n");
open(SETTINGS, "tpcd.runsetup");
#print ("after open settings tpcd.runsetup\n");
&parseVars(SETTINGS);
#print ("after parseVars settings tpcd.runsetup\n");
close (SETTINGS);
#print ("after close settings tpcd.runsetup\n");

sub parseVars{
   local($FILEHANDLE) = @_;
   while (<$FILEHANDLE>){
      chop;
      # strip the export variable name from the setting and the comment
      if (/^(\w*)=([a-zA-Z0-9_\-\+\/\~\\\.:]*)\s*#*(.*)$/){
         $envVar = $1;
         $envSet = $2;
         $ENV{$envVar} = $envSet;
         #print "Setting1 \$ENV{$envVar} to $envSet...\n";
      }
   }
}

# setVar takes a list of environment variables as the first argument and the
value
# that the environment variables will be set to if undefined as the second arg.
# If the second arg is 'ERROR' then the script will end if any of the
environment
# variables in the list is not specified.
sub setVar{
            local $error = 0;
            local @vars = @_;
```

```
	local $val = pop(@vars);
	foreach $var (@vars){
		if ( (length($ENV{$var}) <= 0) || ($ENV{$var} =~ /
null/i) ){
			if($val eq "ERROR"){
				$error = 1;
				print "ERROR: $var must be
defined\n";
			}
			else{
				$ENV{$var} = $val;
			}
		}
	}
	if($error){
		die "make appropriate changes and rerun the script\n";
	}
}
```

macro.pl

```
#!/usr/bin/perl

# 0616 jen Added dirscmd for the simple dir or ls command that just reports
file names

##Heading
#Global Information

# Seed (initialize) random function; not
# required to init in individual functions

srand();

# Set flag to indicate slaves.lst information is not cached
# initialize all slaves.lst arrays to empty

$fvt_slave_list_read = 0;
$fvt_server_read = 0;
@fvt_slave_list = ();
%fvt_slave_list_info = ();
%fvt_servers = ();
%fvt_serverids = ();
%fvt_serverplats = ();

##External
#Platform variables
#
#
#  The following are the supported methods for determing which platform
#  a perl script is running on.  All perl scripts must start with a
#  definition of $AIX = 1 very near the top in order for the conversion
#  of the script to other platforms to properly occur. Note that you
#  test for this after including macro.pl.
#
#  Any UNIX platform:
#     if ($AIX)
#
#  AIX:
#     if ($REAL_AIX)
#
#  AIX 4.1:
#     if ($REAL_AIX4)
#
#  HP:
#     if ($HPUX)
#
#  Sun:
#     if ($SunOS)
#
#  OS/2:
#     if ($OS2)
#
#  SCO OpenServer:
#     if ($SCO_SV)
#
#  SCO UnixWare:
#     if ($UNIX_SV)
#
#  SNI:
#     if ($SINIXN)
#
#  Windows 3.1:
#     if ($WIN)
#
#  Windows NT or Windows 95
#     if ($WINT)
#
#  Windows NT
#     if ($WINT && !$WIN95)
#
#  Windows NT (PPC Flavor)
#     if ($WINTPPC)
#
#  Windows 95:
#     if ($WIN95)
#


# defines for main release and alternate release for jclient

$main_release = "db2_v3";
$alt_release = "db2_v2";


$uname = &get_blduname;
# HPUX and SunOS use AIX scripts.
# WIN95 uses WINT script.
# So we need to use the uname to set the proper variables.
if($uname eq "AIX"){
   $REAL_AIX = 1;
   chop( $aix_ver = `uname -v` );
   if ($aix_ver eq "4")
   {
     $REAL_AIX4 = 1;
   }
}
elsif($uname eq "SunOS"){
   $SunOS = 1;
}
elsif($uname eq "HPUX"){
   $HPUX = 1;
}
elsif($uname eq "SINIXN"){
   $SINIXN = 1;
}
elsif($uname eq "SCO_SV"){
   $SCO_SV = 1;
}
elsif($uname eq "UNIX_SV"){
   $UNIX_SV = 1;
}
elsif($uname eq "WIN95"){
   $WIN95 = 1;
}
elsif($uname eq "WINTPPC"){
   $WINTPPC = 1;
}


##External
#Output buffering
#
# macro.pl will make both STDERR and STDOUT unbuffered as this is
desirable
# to most tools/buckets to guarantee that the output is in chronological order.
# It is also convienient to be able to watch the output as it comes out rather
# than it being buffered in some cases.

&unbuffer(STDOUT,STDERR);



$noPlatFormMsg=
  'Undefined platform or the function is not supported on this platform,
stopped';

$TRUE = 1;
$FALSE = 0;
$rshcmd = "rsh";
```

```
if ( $OS2 )
{
  $dircmd = "dir /s ";
  $dircmds = "dir ";
  $cpcmd = "copy ";
  $cpcmdpostopt = "";
  $cpdircmd = "xcopy ";
  $cpdircmdpreopt = "";
  $cpdircmdpostopt = "/s /e ";
  $cpdirrecpostopt = "/s /e ";
  $cpdirrecpreopt = "";
  $catcmd = "type ";
  $rmcmd = "del ";
  $rmcmdopt= "/n";
  $rmdircmd = "rm -rf ";
  $rncmd = "rename ";
  $mvcmd = "move";
  $redirect= "2>&1";
  $null = "NUL";
  $extractpath=$ENV{SLAVEDRIVE} . "\\build\\extract";
}
elsif ( $WIN )
{
  $dircmd = "dir /s ";
  $dircmds = "dir ";
  $cpcmd = "copy ";
  $cpcmdpostopt = "";
  $cpdircmd = "xcopy ";
  $cpdircmdpreopt = "";
  $cpdircmdpostopt = "/s /e ";
  $cpdirrecpostopt = "/s /e ";
  $cpdirrecpreopt = "";
  $catcmd = "type ";
  $rmcmd = "del ";
  $rmcmdopt= "";
  $rmdircmd = "rd! ";
  $mvcmd = "move";
  $redirect= "";
  $null = "NUL";
  $rncmd = "rename ";
}
elsif ( $WINT )
{
  $dircmd = "dir /s ";
  $dircmds = "dir /b ";
  $cpcmd = "copy ";
  $cpdircmd = "xcopy ";
  $cpdircmdpreopt = "";
  $cpdirrecpreopt = "";
  $catcmd = "type ";
  $mvcmd = "move";
  $null = "NUL";
  $rncmd = "rename ";
  $extractpath=$ENV{SLAVEDRIVE} . "\\build\\extract";

  if(!$WIN95)
  {
    # For Windows NT
    $cpcmdpostopt = "";
    $cpdircmdpostopt = "/e ";
    $cpdirrecpostopt = "/s /e ";
    $rmcmd = "del ";
    $rmcmdopt= "/q";
    $rmdircmd = "rm -rf ";
    $redirect = "2>&1";
    $rncmd = "rename ";
  }
  else
  {
    # For Windows 95
    $cpcmdpostopt = " /y ";
    $cpdircmdpostopt = "/e /y";
    $cpdirrecpostopt = "/s /e /y ";
    $rmcmd = "rm -f ";
    $rmcmdopt= "";
    $rmdircmd = "rm -rf ";
    $redirect = "";
    $rncmd = "rename ";
    $rshcmd = "rexec";
  }
}
elsif ($AIX || $UNIX)
{
  local($PLAT)=&get_uname();

  $dircmd = "ls -ltra ";
  $dircmds = "ls ";
  $cpcmd = "cp ";
  $cpcmdpostopt = "";
  $cpdircmd = "cpdir ";
  $cpdircmdpreopt = "";
  $cpdircmdpostopt = "";
  $cpdirrecpostopt = "";
  $cpdirrecpreopt = "-rp ";
  $catcmd = "cat ";
  $rmcmd = "rm -f ";
  $rmcmdopt = "";
  $rmdircmd = "rm -rf ";
  $mvcmd = "mv";
  $redirect = "2>&1";
  $rncmd = "mv ";
  $null = "/dev/null";

  if ($REAL_AIX)
  {
    $tapecmd = "tctl -f";
  }
  elsif ($SunOS)
  {
    $tapecmd = "mt -f";
  }
  elsif ($HPUX)
  {
    $rshcmd = "remsh";
    $tapecmd = "mt -t";
  }
  elsif ($SCO_SV)
  {
    $rshcmd = "rcmd";
    $tapecmd = "tape -a";
  }
  elsif ($SINIXN)
  {
    $tapecmd = "mt -f";
  }
}
else
{
  print "invalid operating system" ;
  exit 1;
}

%runreport_prefixes = (
  'aixv1',    'c',
  'aixv2',    'c',
  'aixv3',    'csn',
  'hpv2',     'csn',
  'hpv3',     'csn',
  'os2v2',    'a',
  'os2v3',    'os',
  'sunv2',    'csn',
  'sunv3',    'csn',
  'win95v2',  'om',
  'win95v3',  'nm',
  'wintv2',   'b',
  'wintv3',   'nm',
  'winv2',    'wu',
  'winv3',    'wu',
);


%runreport_width = (
  'bucket',  '18',
  'machine', '2',
  'pass',    '4',
  'fail',    '4',
  'defect',  '6',
  'build',   '7',
  'time',    '5',
  'comment', '33',
);

%slaveposn = ('os2',      '40',  #
              'aix',      '42',  #    IMPORTANT !!!!!!
```

```
            'wint',     '44',  #
            'win95',    '46',  #    Make sure all new platforms added go before the
            'hp',       '48',  #    'end' entry.  There is code that searches
            'sun',      '50',  #    'slaveposn' to get out list of supported
            'win',      '52',  #    platforms and it stops when it reaches the 'end'.
            'mvs',      '101',
            'end',      '999',
            'sni',      '80',  #    these are not supported 'yet'
            'sco',      '82',  #

            'abbr',     '24',
            'mpp',      '54',
            'server',   '56',
            'id',       '60',
            'downlv',   '84',
            'srvclass', '86',
            );

@MPP_random_list = ( '_n3', '_n4', '_n5', '_n6', '_n7', '_n8', '_n9', '_n10' );

##Heading
#File, Path and System Macros


##Internal
#buffer
# Given a list of filehandle names, buffer them without
# changing the currently selected filehandle.

sub buffer {
select((s-$-($]%4?'@_=map((':'grep(($_=').'select($_),$l=0),@_)'-ee,@_)
[$[+1]);
}


##External
#cat
#
# macro to concatenate (or type, print) a file.
#
# This macro optionally supports a second parameter that specifies
# a file to redirect the output to (the end of) a file.
# Slashes in both parameters are properly handled.
#
# example:
#    &cat( "test/src/foo.c" );
#    &cat( "test/src/foo.c", "/tmp/out" );
#

sub cat{
  local($src, $dst) = @_ ;

  $src = &sl( $src );

  if ($dst)
  {
    $dst = &sl( $dst );
    system("$catcmd $src >> $dst");
  }
  else
  {
    system("$catcmd $src");
  }
}


##Internal
#chmod
#
# change mode of files in a subdirectory
#

sub chmod {
# do presto and change mode to $mod for all files in $bdir and subdirectories
  local($bdir,$mod) =@_ ;
  local(@subdirs);
  local($rdir);

  if ( $AIX )
  {
    chop ( $rdir = `pwd`);
    chdir $bdir || die "Can't cd to $bdir";
    system("presto *");
    system("rm -rf $bdir/S");
    system("chmod $mod *");
    opendir(D,'.'); local (@f) = grep(!/^\.\.?$/,readdir(D)); closedir(D);

    for (@f)
    {
      if (-d $_) { push(@subdirs,$_); }
    }
    if ($#subdirs >= 0)
    {
      for $sd (@subdirs)
        {
          next if $sd eq '.';
          next if $sd eq '..';
          &chmod("$bdir$delim$sd",$mod);
        }
    }
    chdir $rdir || die "Can't cd to $rdir";
  }
}


##External
#cp
#
# Macro to copy files or directories.
# This macro will properly handle slash conversion in path names
#
# examples:
#
#   &cp( "foo.c", "bar.c" );
#   &cp( "test/src/foo.c", "/tmp" );
#   &cp( "test/src , "/tmp/src" );

sub cp {
  local($src, $dst) = @_ ;
  $src = &sl( $src );
  $dst = &sl( $dst );

  if ( -d $src )
  {
    -d $dst || mkdir ($dst, 0777) || die "Can't mkdir $dst: $!\n";

    # GSM - Windows 95's xcopy chokes on double-slashes in the paths
    if ( $OS2 || ( $WINT && ( !$WIN95 ) ) )
    {
      $src = &sl4( $src );
      $dst = &sl4( $dst );
    }
    &quiet_system("$cpdircmd $cpdircmdpreopt $src $dst $cpdircmdpostopt");
  }
  else
  {
    &quiet_system("$cpcmd $src $dst $cpcmdpostopt");
  }
}


##External
#cprec
#
#  Copy files and directories recursively.
#  This macro will properly handle slash conversion in path names.
#

sub cprec
{
  local($src, $dst) = @_;
  $src = &sl( $src );
  $dst = &sl( $dst );

  -d $dst || mkdir ($dst, 0777) || die "Can't mkdir $dst: $!\n";

  # GSM - Windows 95's xcopy chokes on double-slashes in the paths
  if ( $OS2 || ( $WINT && ( !$WIN95 ) ) )
  {
    $src = &sl4( $src );
    $dst = &sl4( $dst );
  }

  if ($AIX || $UNIX)
```

```
  {
    &quiet_system("$cpcmd $cpdirrecpreopt $src $dst $cpdirrecpostopt");
  }
  else
  {
    &quiet_system("$cpdircmd $cpdirrecpreopt $src $dst $cpdirrecpostopt");
  }
}


##External
#fvt_add_days_to_datestring
#
# Returns a date string (mm-dd-yyyy) which is the sum of the 2 input
parameters:
#
#  $newdate = &fvt_add_days_to_datestring('03-23-1997', '90');
#
#     $newdate would be "06-20-1997".

sub fvt_add_days_to_datestring {

   local( $current, $shiftday ) = @_;
   local( %Tmon, %Tmon2, %Tmon3, $Tyear, $Tday, $Nyear, $Nday,
$Nyear2, $i, $Nmon );

   %Tmon = (1, 0, 2, 31, 3, 59, 4, 90, 5, 120, 6, 151, 7, 181, 8, 212, 9, 243, 10,
273, 11, 304, 12, 334, 13, 365);
   %Tmon2 = (1, Jan, 2, Feb, 3, Mar, 4, Apr, 5, May, 6, Jun, 7, Jul, 8, Aug, 9,
Sep, 10, Oct, 11, Nov, 12, Dec);
   %Tmon3 = (Jan, '01', Feb, '02', Mar, '03', Apr, '04', May, '05', Jun, '06', Jul,
'07', Aug, '08', Sep, '09', Oct, '10', Nov, '11', Dec, '12');

   local($month, $day, $year) = split(/-/, $current);

   $Tyear = ($year % 4);
   $Tday = int($year / 4) * 1461;
   $Tday = $Tday + 365 * ($Tyear - 1);
   $Tday = $Tday - 1 if ($Tyear == 0);
   $Tday = $Tday + $Tmon{$month+0};
   $Tday = $Tday + $day;

   $Tday = $Tday+$shiftday;

   $Nyear = int($Tday / 1461) * 4;
   $Nday = $Tday - $Nyear / 4 * 1461;
   $Nyear = $Nyear - 4 if ($Nday == 0);
   $Nday = 1461 if ($Nday == 0);
   $Nyear2 = int($Nday / 365);
   $Nyear2 = 3 if ($Nyear2 == 4);
   $Nyear = $Nyear + $Nyear2 + 1;
   $Nday = $Nday - $Nyear2 * 365;
   $Nyear = $Nyear - 1 if ($Nday == 0);
   $Nday = 365 if ($Nday == 0);

   for ($i = 13; $i >1; $i--)
   {
         if ($Nday <= $Tmon{$i} + &fvt_Eday($Nyear, $i) &&
            $Nday > $Tmon{$i - 1} + &fvt_Eday($Nyear, $i) )
         {
               $Nmon = $Tmon2{$i - 1} ;
               $Nday = $Nday - ($Tmon{$i - 1} + &fvt_Eday($Nyear, $i));
               last;
         }
   }

   if ($Nday < 10) {
         $Nday = '0' . $Nday;
   }

   $Nyear = $Nyear;
   if ($Nyear < 10) {
         $Nyear = '0' . $Nyear;
   }

   return "$Tmon3{$Nmon}-$Nday-$Nyear\n";

}
##External
#fvt_checkFileLock
#
# function to check wether a file lock exists
#
# Example:
# for ($cnt=0; $cnt < 120 && !&fvt_checkFileLock("file1.lock"); $cnt++)
# { sleep(1);  }
#
# returns 1 if file lock exists, otherwise return 0.
#CAUTION: Avoid infinite loops!!
##

sub fvt_checkFileLock
{
  local ( $filename ) = @_ ;

  if ( -f $filename )
  {
    return 1;
  }
  else
  {
    return 0;
  }
}


##External
#fvt_copy_file_endian
#
#   This routine allows a testcase to be run with binary files in
#   both big endian and little endian format.  At run time, it will
#   copy the proper endian file for the testcase to use.
#
#   If the desired file is "data/testcase.dat", then the two files
#   that are checked into cmvc are "data/testcase.dat.bige" and
#   "data/testcase.dat.littlee".  This macro will copy the appropriate
#   one to "data/testcase.dat".
#
#   Example:
#
#      fvt_copy_file_endian( "data/testcase.dat" );
#

sub fvt_copy_file_endian
{
   local( $file ) = @_;
   local( $e_file );


   if ($AIX && !$SCO_SV)
   {
      $e_file = "$file.bige";
   }
   else
   {
      $e_file = "$file.littlee";
   }

   &cp( $e_file, $file );

}


##External
#fvt_createFileLock
#
# function to create a new file lock
#
# Example:
# if (!&fvt_createFileLock("file1.lock")) { print STDERR "...") }
#
# returns 1 if succeeded to create a new file lock, otherwise return 0.
##

sub fvt_createFileLock
{
  local( $filename ) = @_ ;

  if ( -f $filename )
  {
    return 0;
  }
  open (OUT, ">$filename");
  close(OUT);
```

```
}


##Internal
#fvt_Eday
#
# Used by &fvt_add_days_to_datestring macro.
#
sub fvt_Eday
{
  if ($_[0] % 4 == 0 && $_[1] > 2) {
        1;
  }
  else {
        0;
  }
}


##External
#fvt_file_compare
#
# Returns "1" if files are identical, "0" if not
#
#   Example: $file1 = "file1";
#            $file2 = "file2";
#            $diff = &fvt_file_compare("$file1", "$file2");
#

sub fvt_file_compare {

  local ($file1, $file2, $line1, $line2, $same);

  $file1 = shift || return 0;
  $file2 = shift || return 0;
  $same = 1;

  open (FILE1, "$file1") || return 0;
  open (FILE2, "$file2") || return 0;

  while (<FILE1>) {
    $line1 = $_;
    $line2 = (<FILE2>);
    if ($line1 ne $line2) {
      $same = 0;
      last;
    }
  }
  close(FILE1);
  close(FILE2);
  return $same;
}


##External
#fvt_fmt2sqf
#
# Run the fmt2sqf utility for the specified platform
#
#

sub fvt_fmt2sqf
{
  local( $infile, $outfile ) = @_;
  local( $type ) = "SQLAIX";


  if ($OS2) { $type = "SQLOS2"; }

  &verbose_system( "fmt2sqf $type $infile > $outfile" );
}


##External
#fvt_get_jdbc_portnum
# Returns the jdbc listener port number from the etc/services file
# This macro assumes that the entries in the services file are of the form:
#
#   xregressjdbc   60333/tcp
#   xregressjdbci  60334/tcp
#
# where regress is the userid
# The port number which would be returned in this case is 60333

sub fvt_get_jdbc_portnum
{
   local ($result, $jdbcpat, $services);


   $jdbcpat = "x" . $ENV{USER} . "jdbc";

   $services = &fvt_services_file();

   open (FILE, $services);
   while (<FILE>)
   {
      if (/$jdbcpat/i)
      {
         ( $result ) = /(\d+)/;
         last;
      }
   }

   close (FILE);

   return $result;
}


##External
#fvt_get_smallfs_path
#
#   Syntax:  &fvt_get_smallfs_path();
#
#   Returns a string variable containing the location of a
#   small file system on this system.
#
#   For example:
#
#     $smallfsdir = &fvt_get_smallfs_path();
#
#     would return a string  like "/smallfs" on Unix platforms
#     or "D:" on Intel platforms.
#

sub fvt_get_smallfs_path
{
  local($smallfs);


  if ($AIX)
  {
     $smallfs = "/smallfs";
  }
  else
  {
     $smallfs = $ENV{SMALLFSPATH};
  }
  return $smallfs;
}


##External
#fvt_mkdir_r
#
# Creates directory recursively (does not require higher-level directories to
#  previously exist).  Returns a string containing the directory created if
successful,
#  otherwise nothing.
#
# Example:
#
#     $made = &fvt_mkdir_r("/top/lower/lowest");
#
#       - directories "/top" and "/top/lower" will also be created.
#

sub fvt_mkdir_r
{
  local ( $destpath ) = shift;
  local ( @dirlist, $destroot, $savedir, $curdir, $offroot );

  chop( $savedir = `pwd` );
  chop( $curdir = `pwd` );
```

```
  # The upcoming split will do 2 bad things if the destdir is a
  #  fully-qualified unix path:
  #   1) remove the leading "/"
  #   2) replace it with a blank, which should be shifted off
  # We have to know this will happen so we can fix it later.
  if ( ( $destpath =~ /^\/.*/ ) || ( $destpath =~ /^\w:.*/ ) ) {
    $offroot=1;
  }

  if ( !-d $destpath )   # If destination directory does not yet exist...
  {
    # split destination directory into each subdirectory and make
    # each one that is not there

    @dirlist = split( /\//, $destpath );
    $destroot = shift(@dirlist);

  # The split does 2 bad things if the destdir is fully-qualified:
  #   1) removes the leading "/"
  #   2) replaces it with a blank, which should be shifted off
    if ( $offroot ) {
      if ( @dirlist[0] eq "" )
        {$destroot = shift(@dirlist);
      }
      if ($AIX) {
        $destroot = "/" . $destroot;
      }
      else {
        $destroot = $destroot . "/";
      }
    }

    if (!-d $destroot) {
      mkdir($destroot, 0777);
    }
    chdir( $destroot );
    chop( $curdir = `pwd` );
    for (@dirlist)
    {
      $curdir =~ s/^(.*)\\$/$1/;
      if (!-d "$curdir/$_")
      {
        mkdir( "$curdir/$_", 0777 );
      }
      chdir( "$curdir/$_");
      chop( $curdir = `pwd` );
      $curdir =~ s/^(.*)\\$/$1/;
    }

    if ($offroot == 1) {
      if ( -d "$destpath" )
      {
        $success = "1";
      }
    }
    else {
      if ( -d "$savedir/$destpath" )
      {
        $success = "1";
      }
    }

  # chdir back to where we started

  chdir( $savedir );

  }

  if ($success == 1) {
    return $destpath;
  }
  else {
    return;
  }
}


##External
#fvt_mkfifo
#
# Make a fifo on platforms that support it, else die.
#

sub fvt_mkfifo
{
   local ($fifo) = @_;

   if (&fvt_has_make_fifo)
   {
      &quiet_system ("mkfifo $fifo");
   }
   else
   {
      die "This platform does not support fifos!\n";
   }
}


##External
#fvt_needs_forbit_migration
#
# For migration buckets.  Returns "1" if migration testing of
# "for bit" columns is required.  Else returns "0".
#
sub fvt_needs_forbit_migration {
  if ($REAL_AIX) { return "1"; }
  else { return "0"; }
}


##External
#fvt_nodelock_manipulate
#
# Performs various actions on the current DB2 nodelock file.
# Actions are specified using arguments as in the following examples:
#
#   &fvt_nodelock_manipulate('backup');
#   &fvt_nodelock_manipulate('copynull');
#   &fvt_nodelock_manipulate('copyrel');
#   &fvt_nodelock_manipulate('remove');
#   &fvt_nodelock_manipulate('restore');

sub fvt_nodelock_manipulate {

  local($nodelockdir) = &fvt_get_nodelock_path();
  local ($action, $destination) = @_;

  if ($action eq "backup") {
    if ($AIX) {
      system("dbtsudo nodelock_ctrl backup");
    }
    else {
      &fvt_redirect_output("tempfile.nl");
      &cp("$nodelockdir/nodelock", "$nodelockdir/nodelock.bak");
      &fvt_restore_output();
    }
  }
  if ($action eq "copynull") {
    if ($AIX) {
      system("dbtsudo nodelock_ctrl replace $TESTDIR/nodelock.nul");
    }
    else {
      &fvt_redirect_output("tempfile.nl");
      &cp("$TESTDIR/nodelock.nul", "$nodelockdir/nodelock");
      &fvt_restore_output();
    }
  }
  if ($action eq "copyrel") {
    if ($AIX) {
      system("dbtsudo nodelock_ctrl replace $TESTDIR/nodelock.rel");
    }
    else {
      &fvt_redirect_output("tempfile.nl");
      &cp("$TESTDIR/nodelock.rel", "$nodelockdir/nodelock");
      &fvt_restore_output();
    }
  }
  if ($action eq "remove") {
    if ($AIX) {
      system("dbtsudo nodelock_ctrl delete");
    }
    else {
      &rm("$nodelockdir/nodelock");
```

```
    }
  }
  if ($action eq "restore") {
    if ($AIX) {
      system("dbtsudo nodelock_ctrl restore");
    }
    else {
      &fvt_redirect_output("tempfile.nl");
      &cp("$nodelockdir/nodelock.bak", "$nodelockdir/nodelock");
      &fvt_restore_output();
    }
  }
  return;

}


##External
#fvt_remove_fifo
#
# Remove a fifo that has already been opened.

sub fvt_remove_fifo
{
   local ($fifo) = @_;

   if (&fvt_has_make_fifo)
   {
      &rm($fifo);
   }
   else
   {
      die "This platform does not support fifos!\n";
   }

}


##External
#fvt_removeFileLock
#
# function to remove a file lock
#
# Example:
# if (!&fvt_removeFileLock("file1.lock")) { print STDERR "...") }
#
# returns 1 if succeeded to remove the file lock
##

sub fvt_removeFileLock
{
 local ( $filename ) = @_ ;

 &rm ( $filename );
 return 1;
}


##Internal
#fvt_replace_space_with_underscore
#
# replace spaces in a string with underscore and ! combination
#

sub fvt_replace_space_with_underscore
{
 local($cmdstr) = @_;


 $cmdstr =~ s/ /_!/g;

 return ($cmdstr);
}


##Internal
#fvt_replace_underscore_with_space
#
# replace underscore and ! in a string with spaces
#

sub fvt_replace_underscore_with_space
{
 local($cmdstr) = @_;


 $cmdstr =~ s/_!/ /g;

 return ($cmdstr);
}


##External
#fvt_run_drv
#
# Run a driver file. This function will ensure to run a driver file on all
# platforms.
#
# example:
#   &fvt_run_drv( "setup.drv" );
#

sub fvt_run_drv
{
 local($cmd) = @_ ;


 &verbose_system( &rm_drv_ext( $cmd ) );
}


##External
#fvt_sed
#
# This function acts similarly to the sed command from unix, to replace one
# string with another in a file.
#
# Example:
#    &fvt_sed ("old", "new", "myfile")
#
# will replace all instances of the string "old" with the string "new"
# in the file myfile.

sub fvt_sed
{
   local($old, $new, $file) = @_ ;


   open (file, $file) || die "Can't open file $file";
   open (filenew, ">$file.new") || die "fvt_sed: could not open file ${file}.new";
   print "Replacing \"$old\" with \"$new\" in file $file\n";
   while (<file>)
   {
      s/$old/$new/g;
      print filenew $_;
   }
   close(filenew);
   rename ("$file.new", $file);
}


##Internal
#fvt_services_file
# Returns the fully qualified pathname for the etc/services file
#
sub fvt_services_file
{
   if ($AIX)
   {
      return ("/etc/services");
   }
   elsif ($WINT)
   {
      return ("$ENV{'windir'}/system32/drivers/etc/services");
   }
   elsif ($OS2)
   {
      return ("$ENV{'ETC'}/services");
   }
}


##External
```

```
#fvt_sl
#
# macro to convert single forward slash to double back slashes on OS2
# and single backslash on NT.
#
# example:
#
#     $cmd = &fvt_sl("dothis test/foo.c");
#     system ($cmd);
#

sub fvt_sl
{
  local($src) = @_ ;


  if ( $OS2 )
  {
    $src =~ s:/:\\\\:g;
  }
  elsif ( $WIN || $WINT )
  {
    $src =~ s:/:\\:g;
  }

  return($src);
}


##External
#fvt_system_date
#
# Macro to query or change the system date to a new value.
#
# Usage: &fvt_system_date();
#        &fvt_system_date("mm-dd-yyyy");
#

sub fvt_system_date {

  local($platform) = &get_uname();
  local($newdate) = @_;
  local(@rawstring, $timestring, $month, $day, $year);

  if ($newdate eq "") {
    @rawstring = localtime();
    $timestring = "@rawstring";
    $month = @rawstring[4];
    $month++;              # months are numbered 0 to 11 in perl
    $day = @rawstring[3];
    $year = @rawstring[5];
    $datestring = sprintf ("%02d", $month) ."-". sprintf ("%02d", $day) ."-".
sprintf ("%02d", $year);
  }
  else {
    if ($AIX) {
        &fvt_redirect_output("tempfile.dt");
        system("dbtsudo netls_time_AIX.pl $newdate");
        &fvt_restore_output();
    }
    else {
      system("date $newdate");
    }
  }
  return $datestring;
}


##External
#fvt_tar_and_compress
#
#  Tar and compress a set of files into a single .tar.Z file.
#  NOTE:  The archiveName should be passed without the .tar.Z extension.
#
#  Usage:
#     fvt_tar_and_compress(archiveName, files...)
#
#  Example:
#     fvt_tar_and_compress("v211_se", "SQL00001", "sqldbdir");
#

sub fvt_tar_and_compress
{
  local($archiveName, @archiveFiles) = @_;
  local ($mapfile);
  if (-f $archiveName.tar) { &rm("$archiveName.tar"); }
  if (-f $archiveName.tar.Z) { &rm("$archiveName.tar.Z"); }
  if (-f $mapfile) { &rm("$mapfile"); }

  # On OS/2, apparently all file names are changed to all lowercase,
  #  and empty directories and 0-byte files are excluded.  Solution:
  #  take a snapshot of file names, to use as input for renaming
  #  during uncompress and untar.
  #
  if ($OS2) {
    local ($pwd, $pwdLen);
    local ($source, $emptydir, $srcbak, @sourcedirentries, $file);

    $mapfile = 'fnames.chg';
    chop ($pwd = `pwd`);
    $pwd =~ s/^(\w:)\\$/$1/;              # 'pwd' ends with '\' only in root
    $pwdLen = length($pwd);

    foreach $source (@archiveFiles) {
      opendir (SOURCEDIR, "$source");
      @sourcedirentries = readdir(SOURCEDIR);
      closedir SOURCEDIR;
      if ( $sourcedirentries[$#sourcedirentries] eq "..") {
        $srcbak = $source;
        $source = $source . "\\";
        open (MAP, ">>$mapfile");
        print MAP "$source\n";          # record the names
        close (MAP);
        $source = $srcbak;
        next;
      }
      else
      {
        open (DIRTREE, "dir $source /f /s /o:d |");
        foreach $file (<DIRTREE>) {
          chop $file;
          $file = substr($file, $pwdLen+1);
          print "Archiving:\t$file\n";

          # if it is a directory...
          if ( -d $file ) {
            $emptydir = 1;
            opendir (DIR, "$file");
            local (@direntries) = readdir (DIR);
            foreach (@direntries) {
              if ( ($_ !~ /^\.$/) && ( $_ !~ /^\.\.$/ ) ){
                $emptydir = "0";
              } # if nothing but . and ..
            }
            close (DIR);

            # and it is empty,
            if ($emptydir == "1") {
              $file = $file . "\\";
              open (MAP, ">>$mapfile");
              print MAP "$file\n";       # record the names
              close (MAP);
            }
          }
          elsif ( -z $file )              # or if it is a 0-byte file
          {
            open (MAP, ">>$mapfile");
            print MAP "$file\n";          # record the names
            close (MAP);
          }
          elsif ( $file =~ /[A-Z]/ )      # if any uppercase characters
          {
            open (MAP, ">>$mapfile");
            print MAP "$file\n";          # record the names
            close (MAP);
          }
        }
      }
      close DIRTREE;
    }
  }

  &verbose_system("tar -cf $archiveName.tar $mapfile @archiveFiles");
```

```
  if (-f $mapfile) { &rm("$mapfile"); }

  if ($WINT)
  {
    # Need to force the .Z extension for compatibility with 'compress'
    &verbose_system("gzip -S .Z $archiveName.tar");
  } # if wint
  else
  {
    &verbose_system("compress $archiveName.tar");
  } # else
}


##Internal
#fvt_temporary_has_fork
#
#    TEMPORARY macro for use by the atblb bucket to skip over unix specific
testcases
#    that are currently implemented using fork and are being changed to use
#    fvt_spawn_child.
#

sub fvt_temporary_has_fork
{
  local( $retval ) = 0;


  if ($AIX)
  {
    $retval = 1;
  }

  return $retval;
}


##Internal
#fvt_touch_dir
#
# This macro performs an fvt_touch on all the files and subdirectories in a
# specified directory.  If the recursive mode of this macro is used, this
# macro will traverse all subdirectories of the target directory, fvt_touch'ing
# all files along the way.
#
# Syntax:  fvt_touch_dir( <target directory> [,rec] )
#
# Example:  fvt_touch_dir("test/fvt/standalone/indicat","rec");
#
# Created by: Ryan Sue
# Created on: 07/03/97

sub fvt_touch_dir
{
   local($target, $recnrec) = @_;

   # Check that the first argument passed is a directory.
   if ( !(-d $target) )
   {
      die "ERROR: $target is not a directory!\n";
   }

   print "Beginning fvt_touch_dir...\n";

   # Read the contents of the target directory.
   local(*CONTENTS);
   opendir(CONTENTS,$target) || die "ERROR: Can't read the $target
directory!\n";
   local(@files_dirs) = readdir(CONTENTS);
   closedir(CONTENTS);

   # fvt_touch the files and subdirectories in the target directory.
   # If the recursive mode is used, call fvt_touch_dir for all subdirectories.
   &verbose_system("fvt_touch $target");
   local($file_dir);
   foreach $file_dir (@files_dirs)
   {
      next if $file_dir eq '';
      next if $file_dir eq '.';
      next if $file_dir eq '..';
      if ( (-d "$target/$file_dir") && ($recnrec =~ /rec/i) )
      {
         &fvt_touch_dir("$target/$file_dir","rec");
      }
      &verbose_system("fvt_touch $target/$file_dir");
   }

   print "Finished fvt_touch_dir.\n";

   return 0;
}


##External
#fvt_umask
#
#   performs the umask function for all platforms (where applicable).  This is
#   a perl builtin function, but dies on some platforms so this method is much
#   preferable.
#

sub fvt_umask
{
  local( $mask ) = @_;

  if (!$WINT)
  {
    umask( $mask );
  }
}


##External
#fvt_uncompress_and_tar
#
#   uncompress and untar a file.tar.Z file
#   The filename must be passed in WITHOUT the .tar.Z extension
#
#   Example:
#
#      fvt_uncompress_and_tar( "myfile" );
#

sub fvt_uncompress_and_tar
{
  local( $archfile ) = @_;
  local ($mapfile) = "fnames.chg";
  local ($targetfile, $lowerfile, $targetdir, $lowerdir);


  if ($AIX)
  {
    &verbose_system("uncompress $archfile.tar.Z");
  }
  elsif ($WINT)
  {
    &verbose_system("gzip -d $archfile.tar.Z");
  }
  else
  {
    &verbose_system("compress -d $archfile.tar.Z");
  }

  &verbose_system("tar -xf $archfile.tar");

  # See &fvt_tar_and_compress() for an explanation of the following block
  # The only scenario I know this code does not handle is a 0-byte,
  #  lower-case file, in a directory tree that is also *all* lower-case.
  #  Which I don't think happens.
  if ($OS2) {
    if (-s $mapfile) {
      open (MAP, "<$mapfile");

      foreach $targetfile (<MAP>) {
        chop $targetfile;
        $targetfile =~ tr/\\/\/;          # Transform to UNIX standards for
simplicity

        # Here it gets somewhat ugly... the top directory is OK, but
        # anything under will be lowercase
        $topdir = $targetfile;
        $topdir =~ s/^(\w*)\/(.*)$/$1/;
        ($rest = $2) =~ tr/[A-Z]/[a-z]/;    # Depends on s/// above for $topdir
        $lowerfile = $topdir . "/" . $rest;
```

```
        # Check for a trailing "\", which we used to indicate an empty source
directory
        if ( substr($targetfile, -1, 1)  eq "/" ) {
          &fvt_mkdir_r($targetfile);
          next;
        }

        # Check for 0-byte files
        # if there is not a file or directory by that name... simply 'touch' one
        if ( (!-d $targetfile) && (!-f $targetfile) ) {
          system("touch $targetfile");
          next;
        }

        # Check for mixed- (or upper-) cased names
        # Apparently "-d" and "-f" are not case-sensitive, which sucks.
        # How can I tell whether the current copy (on the disk) is the
        #  right case or not?
        # Perhaps I should not even bother checking... rename everything.
        #    - get top dir (already know it)
        #    - back it up
        $topbak = $topdir;

        #    - get a list of subdirs
        @subdirs = split( /\/|\\/, $targetfile);

        #    - for each subdir
        shift(@subdirs); #don't need the top
        foreach $subdir (@subdirs) {
          # get lowercased version of subdir
          $lowersubdir = $subdir;
          $lowersubdir =~ tr/[A-Z]/[a-z]/;
          $lowersubdir =~ tr/\/\\/;

          # rename unconditionally
          &quiet_system("$rncmd $topdir"."\\"."$lowersubdir $subdir" );

          # append subdir to $topdir and reiterate
          $topdir = $topdir . "\\" . "$subdir";
        }
        $topdir = $topbak;
        next;
      }
      close MAP;
      &rm($mapfile);                    # Users should never see this
    } # if -s $mapfile
  } # if OS/2
}



##External
#fvt_waitFileLock
#
# function to wait on the change of state of a file lock
#
# fvt_waitFileLock( lockname, exists, timeout )
# where
#   exists = 1 if you want to wait for the lock file to exist
#          = 0 if you want to wait for the lock file to not existing
#   timeout is the maximum wait time in seconds
#
# Example:
# if (!&fvt_waitFileLock("file1.lock", 1, 200))
#    {  die "Wait for the existance of file1.lock timed out.\n";  }
#
# returns 0 if request timed out, 1 otherwise
##

sub fvt_waitFileLock
{
  local( $lockname, $exists, $timeout ) = @_ ;
  local( $elapsed );

  for( $elapsed = 0; $elapsed < $timeout && (&fvt_checkFileLock
($lockname) != $exists); $elapsed++ )
     {  sleep(1);  }

  if( &fvt_checkFileLock($lockname) == $exists )
     {  return 1;  }
  else
     {  return 0;  }
}


##External
#get_curdir
#
#   retrieve current directory, will also translate backward slashes
#   to forward so resultant pathname can be used in a platform independant
#   way.
#

sub get_curdir
{
   local($tmp);
   chop($tmp = `pwd`);
   return &rsl($tmp);
}


##External
#mv
#
# macro to move (rename) a file or directory.
#
# This macro moves a file or directory from one location to another.
# Slashes in both parameters are handled.
#
# examples:
#    &mv( "foo.c", "bar.c"  );       # simply renames foo.c to bar.c
#    &mv( "foo.c", "test/src"  );    # moves foo.c to test/src/foo.c
#                                    # if test/src exists,
#                                    # else moves foo.c to *file* test/src
#    &mv( "/usr/bin", "/tmp"  );     # creates /tmp/bin if /tmp exists
#                                    # else creates /tmp


sub mv {
  local($src, $dst) = @_ ;

  $src = &sl( $src );
  $dst = &sl( $dst );

  if ($OS2)
  {
    # if the destination has a drive letter in it, strip this off
    # or else OS/2 will complain (even if it is the same drive)

    # there could still be a problem if the destination drive is different
    # than the source drive, but this is not being addressed until
    # a test bucket actually needs that function

    if (substr( $dst, 1, 1 ) eq ":")
    {
      $dst = substr( $dst, 2 );
    }
  }

  &quiet_system("$mvcmd $src $dst");
}


##External
#regexp_grep
#
#  grep a file for a regular expression - the default grep program
#  on all platforms may not support regular expressions.
#
#  This subroutine is passed two parameters, first the pattern
#  (regular expression pattern) and the second the filename
#  to search in.
#
# Note: NT's grep -e option does not support the [0-9]+ construct.
# e.g.:
#     &regexp_grep('SQL[0-9][0-9]*[A-Z]', 'secur.out');
#     &regexp_grep('DB2[0-9][0-9]*[A-Z]', 'secur.out');
#

sub regexp_grep
{
  local($pattern, $filename) = @_ ;
```

```
  if ( $AIX )
  {
    system ("egrep \"$pattern\" $filename");
  }
  else
  {
    system ("grep -e \"$pattern\" $filename");
  }
}


##External
#rm
#
# macro to remove (delete) a file.
# This macro will properly handle slashes in the specified file name.
#
# syntax:
#   &rm( "test/src/foo.c" );
#

sub rm {
  local($src) = @_ ;

  $src = &sl( $src );
  &quiet_system("$rmcmd $src $rmcmdopt");
}


##Internal
#rm_drv_ext
#
# Removes .drv extension in order to run .drv file on non-unix platforms
#
# example:
#   &rm_drv_ext( "setup.drv" );
#

sub rm_drv_ext
{
  local($cmd) = @_ ;

  if (!$AIX)
  {
    $cmd =~ s/\.drv//;
  }
  return $cmd;
}


##External
#rmdir
#
# Removes a directory AND ALL OF ITS FILES AND SUBDIRECTORIES.
# Slashes in the specified path are handled.
#
# example:
#   &rmdir( "test/src" );
#

sub rmdir {
  local($src) = @_ ;
  $src = &sl( $src );

  # GSM - Windows 95's xcopy chokes on double-slashes in the paths
  if ( $OS2 || ( $WINT && ( !$WIN95 ) ) )
  {
     $src = &sl4( $src );
  }
  &quiet_system("$rmdircmd $src");
}

##External
#rn
#
# macro to rename a file within a given directory.
# Slashes in the specified path are handled properly.
#
# example:
#   &rn( "test/src", "0001.c", "0002.c" );
#

sub rn {
  local($dir,$src,$dst) = @_ ;

  if ($AIX || $UNIX)
  {
     $dst = ( $dir."/".$dst );
  }

  $oldname = &sl( $dir."/".$src );
  $newname = &sl( $dst );

  &quiet_system( "$rncmd $oldname $newname" );
}


##External
#rsl
#
# macro to convert a backslash to a forward slash
#
# example:
#
#     $unix_file = &rsl ("test\foo.c");
#

sub rsl {
  local($src) = @_ ;

  if ( $OS2 || $WIN || $WINT )
  {
     $src =~ tr/\\/\/s ;
  }
  return($src);
}


##External
#sl
#
# macro to translate forward slashes to reverse slashes if this
# happens to be running on an intel platform.
#
# This is necessary prior to using any path names in system functions
# and allows the path names to be specified in unix format in driver
# files
#
# example:
#
#    $file = &sl( "test/foo.c" );
#    system( "myprogram $file" );
#
# This is also essential for setting environment variables.
#
# example:
#
#    $ENV{'SOMEPATH'} = &sl ("test/foo");
#

sub sl {
  local($src) = @_ ;

  if ( $OS2 || $WIN || $WINT )
  {
    $src =~ tr/\/\\/s ;
  }
  return($src);
}


##External
#sl4
#
# macro to convert single back slashes to double back slashes.
#
# This is required for most system calls, and is often used in
# combination with the &sl macro.
#
# example:
#
#     $cmd = &sl4 ( &sl("dothis test/foo.c") );
#     system ($cmd);
#
```

```
sub sl4 {
  local($src) = @_ ;

  if ( $OS2 || $WIN || $WINT )
  {
    $src =~ s/\\/\\\\/g ;
  }
  return($src);
}


##Internal
#unbuffer
# Given a list of filehandle names, unbuffer them without
# changing the currently selected filehandle.

sub unbuffer {
select((s-$-($]%4?'@_=map((':'grep(($_=').'select($_),$l=1),@_)'-ee,@_)
[$[+1]);
}


##Heading
#NLS Macros


##Internal
#change_codepage
#
# This macro is used to switch current operation from being in one codepage
to
# being in another.  This macro can only be used on the following platforms:
# OS/2, Windows NT, and Windows 3.1.
#
# syntax:  &change_codepage( CPAGE );
#
# In the syntax above, CPAGE is the codepage (numerical) you wish to start
# operating in.
#

sub change_codepage {
  local($src)=@_ ;

  if( $OS2 || $WIN || $WINT )
  {
    system("chcp $src");
  }
}


##Internal
#fvt_get_codepage_list
#
# This macro returns a list (string) containing the codepages that are
# valid on the current platform for the language (AIX locale name format)
# given.  The elements of the list are separated by commas.
#
# syntax:  &fvt_get_codepage_list(LANG);
#
# ex/  Assuming the platform currently being used is
#

sub fvt_get_codepage_list
{
   local($srclang)=@_;
   local(%cplist);

   if ($REAL_AIX)
   {
      %cplist = ( 'Ja_JP', 'Ja_JP,932,ja_JP,954',
                  'Zh_TW', 'Zh_TW,950,zh_TW,964',
                  'zh_CN', 'zh_CN,1383',
                  'ko_KR', 'ko_KR,970',
                 );
   }
   elsif ($HPUX)
   {
      %cplist = ( 'Ja_JP', 'ja_JP.SJIS,5039,ja_JP.eucJP,954',
                  'Zh_TW', 'zh_TW.big5,950,zh_TW.eucTW,964',
# Temporarily removed until DB2 supports this locale name.
#                  'zh_CN', 'zh_CN.hp15CN,1383',
                  'ko_KR', 'ko_KR.eucKR,970',
                 );
   }
   elsif ($SunOS)
   {
      %cplist = ( 'Ja_JP', 'ja,954',
                  'Zh_TW', 'big5,950,zh_TW,964',
                  'zh_CN', 'zh,1383',
                  'ko_KR', 'ko,970',
                 );
   }
   elsif ($OS2)
   {
# Removed 942 and 943 from Ja_JP - not installed on slaves.
# Removed 938 and 948 from Zh_TW - not installed on slaves.
      %cplist = ( 'Ja_JP', '-,932',
                  'Zh_TW', '-,950',
                  'zh_CN', '-,1381',
                  'ko_KR', '-,949',
                 );
   }
   elsif($WINT)
   {
      %cplist = ( 'Ja_JP', '-,943',
                  'Zh_TW', '-,950',
                  'zh_CN', '-,1381',
                  'ko_KR', '-,949',
                 );
   }
   return($cplist{$srclang});
}


##External
#fvt_is_wchartype_convert_supported
#
#returns true if wchartype conversion is supported
#

sub fvt_is_wchartype_convert_supported
{
   if (!$WINT)
   {
     return 1;
   }
   return 0;
}


##External
#fvt_set_wchart
#
# This macro stores into the environment variable, C_FLAGS_PRE, the
appropriate
# compiler options to enable wchar_t support.  The compiler options used will
# depend on the platform the macro is called from.  The C_FLAGS_PRE
# environment variable is used by the rtest tool to determine what options
# to use during testcase compilation.
#
# This macro also stores into the environment variable, C_LIBS_PRE, the
appropriate
# linker options to enable wchar_t support.  The linker options used will
# depend on the platform the macro is called from.  The C_LIBS_PRE
# environment variable is used by the rtest tool to determine what options
# to use during testcase linking.
#
# syntax:  &fvt_set_wchart();
#

sub fvt_set_wchart
{
   if ( $REAL_AIX )
   {
      $ENV{'C_FLAGS_PRE'}="-qmbcs ";
   }

   if ( $OS2 )
   {
      $ENV{'C_FLAGS_PRE'}="/Sn ";
   }

   if ( $SunOS )
```

```
   {
      $ENV{'C_LIBS_PRE'}="-lw ";
   }
}


##External
#fvt_set_dbcs
#
# This macro stores into the environment variable, C_FLAGS_PRE, the
appropriate
# compiler options to enable DBCS support.  The compiler options used will
# depend on the platform the macro is called from.  The C_FLAGS_PRE
# environment variable is used by the rtest tool to determine what options
# to use during testcase compilation.
#
# syntax:  &fvt_set_dbcs();
#

sub fvt_set_dbcs
{
   if ( $REAL_AIX || $SunOS )
   {
      $ENV{'C_FLAGS_PRE'}="-qmbcs ";
   }

   if ( $OS2 )
   {
      $ENV{'C_FLAGS_PRE'}="/Sn+ ";
   }

   if ( $HPUX )
   {
      $ENV{'C_FLAGS_PRE'}="-Y ";
   }
}


##Internal
#get_db_langs
#

sub  get_db_langs
{

  local( $type ) = @_ ;

  &get_sysval( );

  if ($type eq "") { $type = substr( $sysval{'suffix'}, 1 ); }


  @locales = ('Ja_JP','ko_KR','zh_CN','Zh_TW');

  if ($type eq "win")   { @locales = (); }
  if ($type eq "win95") { @locales = (); }
  if ($type eq "hp")    { @locales = ('ko_KR','Zh_TW'); }
  if ($type eq "sun")   { @locales = ('ko_KR','zh_CN'); }

  return @locales;
}


##Internal
#get_syslang
#
# This macro determines what platform is currently being used, and then
returns
# an associative array containing the locales available on the platform.  The
# keys of the array are the AIX names for the locales, and the values of the
# array are the names for the corresponding locales associated with the
# currently used platform.
#
# syntax:  &get_syslang();
#
# ex/  Assuming the platform currently being used is SINIX, the following
array
#      would be returned by the macro:
#                %syslang = ( 'En_US', 'En',
#                             'da_DK', 'De',
#                             'fr_FR', 'Fr',
#                             'es_ES', 'Es',
#                             'it_IT', 'It',
#                             'ko_KR', 'Kr', );
#

sub get_syslang
{
  local( %syslang );


  %syslang = ( 'En_US', 'En_US',
               'en_US', 'en_US',
               'da_DK', 'da_DK',
               'fr_FR', 'fr_FR',
               'es_ES', 'es_ES',
               'it_IT', 'it_IT',
               'ja_JP', 'ja_JP',
               'Ja_JP', 'Ja_JP',
               'ko_KR', 'ko_KR',
               'zh_CN', 'zh_CN',
               'zh_TW', 'zh_TW',
               'Zh_TW', 'Zh_TW',
               );

  if ($OS2)
  {
    $syslang{"En_US"} = "ENUS437";
    $syslang{"en_US"} = "ENUS437";
  }
  elsif($WINT)
  {
    %syslang = ( 'En_US', 'english_us',
                 'da_DK', 'danish_denmark',
                 'fr_FR', 'french_france',
                 'es_ES', 'spanish_spain',
                 'it_IT', 'italian_italy',
                 'Ja_JP', 'japanese_japan',
                 'ko_KR', 'korean_korea',
                 'zh_CN', 'chinese-simplified_china',
                 'Zh_TW', 'chinese-traditional_taiwan',
                 );
  }
  elsif($HPUX)
  {
    %syslang = ( 'En_US', 'en_US.iso88591',
                 'da_DK', 'da_DK.iso88591',
                 'fr_FR', 'fr_FR.iso88591',
                 'es_ES', 'es_ES.iso88591',
                 'it_IT', 'it_IT.iso88591',
                 'ja_JP', 'ja_JP.eucJP',
                 'ko_KR', 'ko_KR.eucKR',
                 'zh_TW', 'zh_TW.eucTW',
                 'Zh_TW', 'zh_TW.big5',
                 'zh_CN', 'zh_CN.hp15CN',
                 );
  }
  elsif( $SunOS )
  {
    %syslang = ( 'En_US', 'en_US',
                 'fr_FR', 'fr',
                 'es_ES', 'es',
                 'it_IT', 'it',
                 'ko_KR', 'ko',
                 'ja_JP', 'ja',
                 'zh_CN', 'zh',
                 'Zh_TW', 'big5',
                 'zh_TW', 'zh_TW',
                 );
  }
  elsif( $SINIX )
  {
    %syslang = ( 'En_US', 'En',
                 'fr_FR', 'Fr',
                 'es_ES', 'Es',
                 'it_IT', 'It',
                 'ko_KR', 'Kr',
                 );
  }
  elsif( $SCO_SV )
  {
    %syslang = ( 'En_US', 'en_US',
                 'fr_FR', 'fr_FR',
                 'es_ES', 'es_ES',
```

```
            );
  }

  elsif( $UNIX_SV )
  {
    %syslang = ( 'en_US', );
  }

  return %syslang;
}


##External
#LANG_exists
#
# This macro determines if a given LANG (locale) exists on a given platform.
#
# syntax:  &LANG_exists( LANGUAGE );
#
# In the syntax above, LANGUAGE is the name of the locale in question
# (ex. Ja_JP)
# know if the given LANG exists on.  If the given LANG exists on the current
# platform, the macro returns '1', otherwise, the macro returns '0'.
#

sub LANG_exists
{
  local( $src ) = @_ ;


  %syslang = &get_syslang;
  if( $syslang{ $src } )
  {
    return ( 1 );
  }
  else
  {
    return ( 0 );
  }

}


##External
#query_LANG
#
# This macro takes no arguments and returns the name of the locale (AIX
version
# of the name) currently being used.
#
# syntax:  &query_LANG();
#

sub query_LANG
{
  local($PLAT) = &get_blduname();


  %syslang = &get_syslang;
  while(($IBM_lang_name, $OEM_lang_name) = each %syslang)
  {
    if($ENV{LANG} eq $OEM_lang_name)
    {
      return $IBM_lang_name;
    }
  }

  return "En_US";
}


##External
#set_LANG
#
# This macro changes the current setting of the environment variable LANG to
the
# setting given as the parameter to the macro.
#
# syntax:  &set_LANG( LANGUAGE );
#
# In the syntax above, LANGUAGE is the name of the locale (AIX name) (ex.
Ja_JP)
# that you would like to set LANG to.
# This macro returns '1' if LANG was successfully set, and returns '0'
# otherwise.
#

sub set_LANG
{
  local( $src ) = @_ ;
  local( $lang_exists ) = 1;
  local( $lang );


  %syslang = &get_syslang;
  $lang = $syslang{ $src };
  if( $lang eq "" )
  {
    $lang_exists = 0;
  }
  else
  {
    $ENV{'LANG'} = $lang;
  }

  return ( $lang_exists );
}


##External
#set_LC_ALL
#
# This macro changes the current setting of the environment variable LC_ALL
to
# the setting given as the parameter to the macro.
#
# syntax:  &set_LC_ALL( LANGUAGE );
#
# In the syntax above, LANGUAGE is the name of the locale (AIX name) (ex.
Ja_JP)
# that you would like to set LC_ALL to.
# This macro returns '1' if LC_ALL was successfully set, and returns '0'
# otherwise.
#

sub set_LC_ALL
{
  local( $src ) = @_ ;
  local( $lang_exists ) = 1;
  local( $lang );

  %syslang = &get_syslang;
  $lang = $syslang{ $src };
  if( $lang eq "" )
  {
    $lang_exists = 0;
  }
  else
  {
    $ENV{'LC_ALL'} = $lang;
  }

  return ( $lang_exists );
}


##External
#set_LC_CTYPE
#
# This macro changes the current setting of the environment variable
LC_CTYPE to
# the setting given as the parameter to the macro.
#
# syntax:  &set_LC_CTYPE( LANGUAGE );
#
# In the syntax above, LANGUAGE is the name of the locale (AIX name) (ex.
Ja_JP)
# that you would like to set LC_CTYPE to.
# This macro returns '1' if LC_CTYPE was successfully set, and returns '0'
# otherwise.
#

sub set_LC_CTYPE
{
```

```
 local( $src ) = @_ ;
 local( $lang_exists ) = 1;
 local( $lang );

 %syslang = &get_syslang;
 $lang = $syslang{ $src };
 if( $lang eq "" )
 {
   $lang_exists = 0;
 }
 else
 {
   $ENV{'LC_CTYPE'} = $lang;
 }

 return ( $lang_exists );
}


##External
#set_LC_MESSAGES
#
# This macro changes the current setting of the environment variable LANG to
the
# setting given as the parameter to the macro.
#
# syntax:  &set_LC_MESSAGES( LANGUAGE );
#
# In the syntax above, LANGUAGE is the name of the locale (AIX name) (ex.
Ja_JP)
# that you would like to set LANG to.
# This macro returns '1' if LANG was successfully set, and returns '0'
# otherwise.
#

sub set_LC_MESSAGES
{
 local( $src ) = @_ ;
 local( $lang_exists ) = 1;
 local( $lang );


 %syslang = &get_syslang;
 $lang = $syslang{ $src };
 if( $lang eq "" )
 {
   $lang_exists = 0;
 }
 else
 {
   $ENV{'LC_MESSAGES'} = $lang;
 }

 return ( $lang_exists );
}


##External
#fvt_is_dbcs
#
# Returns TRUE if it currently DBCS otherwise returns false.
#

sub fvt_is_dbcs
{
 return $ENV{FVT_IS_DBCS};
}



##Heading
#Client Server Macros


##External
#Client Server variables
#
#
#  The following environment variables are set by the regression tools
#  when client/server buckets are run.   They are the only environment
#  variables outside of the driver file that may be set, and will need
#  to be set manually if the bucket is run manually.
#
#
****************************************************************
**
#
#  If the regression tool cs_run is used to run the bucket, it will set all
#  the necessary environment variables
#
#
****************************************************************
**
#
#  CLIENT_SERVER - set as 'defined' if we are running in client server mode
#  CS_SERVER     - set to the name of the server machine
#  CS_BUCKET     - name of the bucket
#  CS_SERVICE    - set as 'x'.$ENV{USER} (set in macro.pl, no need to set
manually)
#
#  The following environment variables control different modes of
client/server
#  testing.  Only ONE of them may be set at a time.
#
#  LOOPBACK      - running in loopback mode
#  MVSAPPC       - running ddcs to a MVS server - SNA connection
#  MVSTCPIP      - running ddcs to a MVS server - TCP/IP connection
#
#  For MVS connections, the MVS database must also be specified.  The
#  environment variable FVT_MVS_TARGETDB is used to specify this.
#
#  For SPM connections, the regression tools assume the luname of the SPM
#  connection to be defined as SPMLUNAME.  There is no mechanism to
override
#  this.
#
#  There are cases where testcases want to use the platform independance of
the
#  createdb macro, but in client/server mode need to do the cataloging
themselves.
#  To support this case, the following environment variable is supported by the
#  createdb and attach macros and will cause them to issue the create command
#  as if the operation was local, letting the caller control the attach
#
#  IGNORE_CLIENTSERVER - issue commands as if local (for create and
attach macros)
#
#  Please note:  Either in regression or on your own development machines,
#             the client/server environment requires the client to be able
#             to rsh to the server, at the very least to reset the dbm config
#             and do a db2start at the server.  This is all handled by the
#             client/server tools, but requires that the server machine have the
#             client machine in the equivalent of its .rhosts file.
#

if (&is_client_server( ))
{
  $ENV{CS_SERVICE}='x'.$ENV{USER};
  $appcnodename = "APCNODE";
  $tcpipnodename = "TCPNODE";
  $usr_pwd=&get_password;
}
$spmluname = "SPMLUNAM";


##External
#attach
#
# Attach to the server.
# (do this if you need to create databases or update dbm cfg, etc..)
# Dont forget to call &detach when you are done.
# &attach takes as arguments the list of commands to execute on the
# server while attached.
#
# Note: &attach causes the remote node to be catalogued automatically.
#
# Note: The IGNORE_CLIENTSERVER environment variable will cause this
macro
#       to execute the command locally
#
# Example: &attach("create db mydb",
#                  "update dbm cfg for mydb using authentication server");

sub attach
```

```
{
   local(@db2cmnd)=@_;
   local($node)="REMNODE";
   local($nodetype);


   if (!defined $ENV{CLIENT_SERVER} || $ENV
{IGNORE_CLIENTSERVER} )
   {
      &db2(@db2cmnd);
      return;
   }

   &catnode("$node");
   &db2("attach to $node user ".$ENV{USER}." using ".$usr_pwd,
       @db2cmnd );
}


##External
#catnode
#
#   Catalog a node.
#
#   Example:
#      &catnode( "remote_note" );
#

sub catnode
{
   local($node)=@_;


   if(!$node){ $node = "REMNODE"; }

   if (!defined $ENV{CLIENT_SERVER}) {return;}

   &db2("UNCATALOG NODE $node",
       "CATALOG TCPIP NODE $node REMOTE $ENV{'CS_SERVER'}
SERVER $ENV{'CS_SERVICE'}",
       "terminate" );
}


##External
#db2start
#
# Start the database manager localy in standalone or remotely
# in a client-server environement.

sub db2start
{
   if (&is_client_server( ))
   {
      &remote_shell_path( $ENV{CS_SERVER}, "db2start");
   }
   else
   {
      &quiet_system("db2start");
   }
}


##External
#db2stop
#
# Stop the database manager localy in standalone or remotely
# in a client-server environement.

sub db2stop
{
   if (&is_client_server( ))
   {
      &remote_shell_path( $ENV{CS_SERVER}, "db2stop");
   }
   else
   {
      &quiet_system("db2stop");
   }
}
```

```
##External
#detach
#
# Detatch from the server.
# Call this function after calling &attach().
sub detach
{
   if (!defined $ENV{CLIENT_SERVER}) {return;}
   if (! $WIN)
   {
      &db2("detach", "terminate");
   }
}


##External
#dropdb
#
#   Drop a database and optionally it's alias.
#
#   Example:
#
#      &dropdb( "alias", "database" );
#

sub dropdb
{
   local($dbalias,$dbname)=@_;


   if ( $dbname eq "" )
   {
     $dbname=$dbalias;
     if (&is_loopback( ) )
     {
       $dbname = substr( "r$dbalias", 0, 8 );
     }
   }

   if (&is_client_server( ))
   {
      &db2("uncatalog database $dbalias", "terminate");
      if (!&is_ddcs() )
      {
        &remote_shell_path( $ENV{CS_SERVER}, "db2 -v drop database
$dbname" );
      }
   }
   else
   {
      &db2("drop database $dbname", "terminate");
   }
}


##External
#fvt_catalog_spm
#
# catalog spm as a database
#
# Example: &fvt_catalog_spm( "tcpip", "spmdb" );
#

sub fvt_catalog_spm
{
  local($protocol, $spmalias) = @_;


# &quiet_system( "db2 catalog database $spmluname as $spmalias at node
$tcpipnodename" );
  &db2( "catalog database $spmluname as $spmalias at node
$tcpipnodename", "terminate" );
}


##External
#fvt_create_catalog_DRDA_db
#
# create a db on the server and catalog locally through DRDA
# It takes 4 arguments: protocol used, db alias, db name, configs.
# If db name is ommitted, db alias is used for both alias and db name
#
```

```
# Examples:  &fvt_create_catalog_DRDA_db( "tcpip", "testdb" );
#

sub fvt_create_catalog_DRDA_db
{
  local($protocol, $dbalias, $dbname, $configs) = @_;


  if ($dbname eq "" )
  {
    $dbname = $dbalias;
  }
  &fvt_cs_server_system("db2 create database $dbname $configs
authentication client");

  if ($protocol eq "tcpip")
  {
    &db2( "catalog tcpip node $tcpipnodename remote $ENV{CS_SERVER}
server 446", "terminate");
    $nodename = $tcpipnodename;
  }
  elsif ($protocol eq "appc")
  {
    &db2( "catalog appc node $appcnodename remote $ENV{CS_SERVER}
security program", "terminate");
    $nodename = $appcnodename;
  }

  &db2( "catalog db $ENV{CS_SERVER} as $dbalias at node $nodename
authentication dcs",
        "catalog dcs db $ENV{CS_SERVER} as $dbname",
        "terminate" );
}


##External
#fvt_create_catalog_remote_db
#
# create a remote db and catalog a local alias to it
# It takes 4 arguments: protocol used, db alias, db name, configs.
# If db name is ommitted, db alias is used for both alias and db name
#
# Example: &fvt_create_catalog_remote_db("tcpip","testdb","rtestdb","using
codeset ISO8859-1 territory fr_FR");
#          &fvt_create_catalog_remote_db("tcpip","testdb");
#

sub fvt_create_catalog_remote_db
{
  local($protocol, $dbalias, $dbname, $configs) = @_;


  $nodename = &fvt_get_catnode_name($protocol);

  &db2( "catalog $protocol node $nodename remote $ENV{CS_SERVER}
server $ENV{CS_SERVICE}", "terminate");

  if (&is_loopback( ) )
  {
    if ($dbname eq "" )
    {
      $dbname = substr( "r$dbalias", 0, 8 );
    }
    &db2( "create database $dbname $configs authentication client",
          "catalog database $dbname as $dbalias at node $nodename",
          "terminate");
  }
  else
  {
    if ($dbname eq "" )
    {
      $dbname = $dbalias;
    }
    &fvt_cs_server_system("db2 create database $dbname $configs
authentication client");
    &db2("catalog database $dbname as $dbalias at node $nodename
authentication client", "terminate");
  }
}


##External
#fvt_create_tm_db
#
# create a db locally as a tm transaction db
#
# Example: &fvt_create_tm_db( "dbname" );
#

sub fvt_create_tm_db
{
  local($dbname) = @_;


# &quiet_system("db2 create database $dbname");
# &quiet_system("db2 update dbm cfg using tm_database $dbname");
  &db2( "create database $dbname",
        "update dbm cfg using tm_database $dbname",
        "terminate" );
}


##External
#fvt_cs_server_system
#
# Execute a system command on the server defined by $ENV{CS_SERVER}.
# Non-client server or loopback runs will execute the command locally.
#
# Note that this macro will allow commands like "setup.drv" to be executed
# and will handle them properly either locally or on the server.
#
# Example: &fvt_cs_server_system( "db2start" );
#

sub fvt_cs_server_system
{
  local($cmd) = @_;


  # if real client server - execute command on host

  if (&is_client_server( ) && !&is_loopback( ))
  {
    if ($serverplats{$ENV{FVT_JOB_CLASS}} eq "MVS" )
    {
      die "Executing a command on an MVS server is not permitted - bucket
terminated\n";
    }

    $cmd = &fvt_replace_space_with_underscore($cmd);
    &remote_shell_path( $ENV{CS_SERVER}, "tstdirdo $ENV
{CS_BUCKET} $ENV{USER} $cmd");
  }
  else
  {
    # loopback or standalone - execute it here

    &fvt_run_drv( $cmd );
  }
}


##External
#fvt_cs_restart_server
#
# Execute the appropriate command to restart the DB2 server.  On most
servers,
# this will simply execute the fvt_cs_server_system command to do a db2start,
# however for MVS servers this will do nothing.
#
# Example: &fvt_cs_restart_server( );
#

sub fvt_cs_restart_server
{

  if ($serverplats{$ENV{FVT_JOB_CLASS}} ne "MVS" )
  {
    &fvt_cs_server_system( "fvt_db2start_comm" );
  }

}
```

```
##Internal
#fvt_ddcs_catalog_db
#

sub fvt_ddcs_catalog_db{
  local($dbalias, $configs) = @_;
  local( $nodename );

  if ($ENV{MVSAPPC})
  {
#    &quiet_system("db2 catalog appc node $appcnodename remote $ENV
{CS_SERVER} security program");
    &db2( "catalog appc node $appcnodename remote $ENV{CS_SERVER}
security program", "terminate");
    $nodename = $appcnodename;
  } else {
#    &quiet_system("db2 catalog tcpip node $tcpipnodename remote $ENV
{CS_SERVER} server 446");
    &db2( "catalog tcpip node $tcpipnodename remote $ENV{CS_SERVER}
server 446", "terminate");
    $nodename = $tcpipnodename;
  }

# &quiet_system("db2 catalog db $ENV{CS_SERVER} as $dbalias at node
$nodename authentication dcs");
# &quiet_system("db2 catalog dcs db $ENV{CS_SERVER} as $ENV
{FVT_MVS_TARGETDB}");
  &db2( "catalog db $ENV{CS_SERVER} as $dbalias at node $nodename
authentication dcs",
       "catalog dcs db $ENV{CS_SERVER} as $ENV
{FVT_MVS_TARGETDB}",
       "terminate" );
}


##Internal
#fvt_get_catnode_name
#
# returns the nodename used to catalog node, according to the protocol being
used
#
# Examples: &fvt_get_catnode_name("tcpip") returns $tcpipnodename
#          &fvt_get_catnode_name("appc")  returns $appcnodename
#

sub fvt_get_catnode_name
{
  local($protocol) = @_;

  if ($protocol eq "tcpip")
  {
    return $tcpipnodename;
  }
  elsif ($protocol eq "appc")
  {
    return $appcnodename;
  }
}


##External
#fvt_switch_server
#
#   Switches to one of 4 predefined servers in a client/multiple server
# environment.  The multiple servers must have been set up using the
# /server2:.... /server3:.... options of cs_run.
#
# Currently, this only supports all servers using tcp/ip.
#
# sample usage:
#
#   &fvt_switch_server( 3 );
#

sub fvt_switch_server
{
  local( $servernum ) = @_;
  local( $servervar );


  if ($servernum < 1 || $servernum > 4)
  {
    printf( "Server number $servernum out of valid range 1 to 4\n" );
    return;
  }

  $temp = sprintf( "CS_SERVER%1.1d", $servernum );
  &debug_printf( "switching to $temp" );
  $ENV{CS_SERVER} = $ENV{$temp};
  $temp = sprintf( "APCNODE%1.1d", $servernum );
  $appcnodename = $temp;
  $temp = sprintf( "TCPNODE%1.1d", $servernum );
  $tcpipnodename = $temp;

}


#External
#fvt_set_t3path
#
# This macro sets up the operating environment to access T3, and also
# sets an environment variable T3PATH to point to T3 for purposes of
# obtaining the bind files in T3
#

sub fvt_set_t3path
{
  local( $t3basepath, $t3version, $pathsep );


  if ($ENV{T3PATH})
  {
    printf( "T3 already set to $ENV{T3PATH}, will not be reset\n" );
    return;
  }

  if ($AIX)
  {
    $pathsep = ":";
    $t3basepath = "/svtbin/TOOLS/T3/cur";
  }
  else
  {
    $pathsep = ";";
    $t3basepath = "v:/t3";
  }

  if ($REAL_AIX)
  {
    $t3version = "AIX3-CS21";
  }
  elsif ($SunOS)
  {
    $t3version = "SUN-CS21";
  }
  elsif ($HPUX)
  {
    $t3version = "HP-CS21";
  }
  elsif ($SINIXN)
  {
    $t3version = "SNI-CS21";
  }
  elsif ($SCO_SV)
  {
    $t3version = "SCO-CS21";
  }
  elsif ($OS2)
  {
    $t3version = "OS2-CS21";
  }
  elsif ($WINT)
  {
    $t3version = "NT-CS21";
  }
  else
  {
    printf( "T3 not available for this platform\n" );
    exit 0;
  }

  $ENV{T3PATH} = &sl( "$t3basepath/$t3version/" );
  $ENV{PATH} = "$ENV{PATH}$pathsep$ENV{T3PATH}$pathsep";
```

```
}


# This macro terminates any 'mondb' running on this id.
sub fvt_terminate_mondb
{
  local($userid,$PID, $PPID, $myC , $myStime, $myTty, $myTime, $myCmd,
$myCmd2);
  local($name,$KILLPID, $PARPID, $user);

  if ($HPUX || $SunOS)
  {
    open(PIDFILE,"ps -fu $ENV{'USER'} | grep mondb | grep -v grep |");
    while (<PIDFILE>)
    {
      /^ *(.*)/;      # remove leading blanks
      ($userid, $PID, $PPID, $myC , $myStime, $myTty, $myTime, $myCmd, }
$myCmd2) = split (" +", $1);
      print "$PID: $userid:$myCmd\n";
      open(PIDFIL2,"ps -e | grep $PID |");
      while (<PIDFIL2>)
      {
        ($name, $KILLPID, $PARPID, $myC , $myStime, $myTty, $myTime,
$user, $myCmd2) = split(" +", $1);
        print "$KILLPID: $PARPID\n";
        kill( 9, $KILLPID);
      }
      close(PIDFIL2);
      kill(9, $PID);
    }
    close(PIDFILE);
  }
  elsif ($REAL_AIX)
  {
    open(PIDFILE,"ps -e -F'%p:%u:%a' | grep mondb | grep -v grep | grep
ENV{'$USER'} |");
    while (<PIDFILE>)
    {
      print $_;
      ($PID, $name, $user) = split(/:/);
      open(PIDFIL2,"ps -e -F'%p:%P' | grep $PID |");
      while (<PIDFIL2>)
      {
        print $_;
        ($KILLPID, $PARPID) = split(/:/);
        kill( 9, $KILLPID);
      }
    }
    close(PIDFIL2);
    kill(9, $PID);
  }
}



# This macro determines if dereferencing a NULL pointer generates a
SIGSEGV
# (segmentation fault).  On HP, for instance, dereferencing a NULL pointer
# generates a SIGBUS, instead of a SEGV as on SUN, AIX, SCO, and SNI
sub fvt_deref_null_segv
{
   local($rc);
   $rc = 1;
   if ($REAL_AIX || $SunOS)
   {
     # return TRUE
     $rc = 1;
   }
   elsif ($HPUX || $SCO_SV || $SINIXN)
   {
     # return FALSE
     $rc = 0;
   }
   return ($rc);
}

# This macro get a compiler option to place the string literals in read-only area
# of (e.g. used in a1052 test bucket to test for signal handling)
sub fvt_set_readonly_string_compileflag
{
   local($compflag);
   if (($REAL_AIX) || ($SunOS))
   {
     $compflag=" -qro ";
   }
   elsif ($HPUX)
   {
     $compflag=" +T -Wb,+ESlit -Wb,+u1 ";
   }
   elsif ($SINIXN)
   {
     $compflag=" -Krostr ";
   }
   elsif ($SCO_SV)
   {
     $compflag=" -rodata ";
   }
   return($compflag);


#External
#fvt_get_tapename
#
# This macro get the tapename by index.  The index is 0-based (e.g tape drive
0).
# The arguments for this macro is $tapenum (tape number) and $no_rewind
(=0 if
# we want the tape to rewind automatically when end of tape is reached).
# For example, fvt_get_tapename(0,0) means tape drive 0 and the tape will
rewind
# automatically when end of tape is reached. fvt_get_tapename(2,1) means we
get
# the name for tape drive 2 and do not rewind when end of tape is reached.
# Note that this macro does not check whether or not a specified tape device is
# physically connected; it merely get the name syntax of the tape drive.

sub fvt_get_tapename
{
   local($tapenum,$no_rewind)=@_;
   local($tapename) = "";

   if ($SunOS)
   {
     if ($no_rewind)
     {
       $tapename = "/dev/rmt/$tapenum"."n";
     }
     else
     {
       $tapename = "/dev/rmt/$tapenum";
     }
   }
   elsif ($HPUX)
   {
     if ($no_rewind)
     {
       $tapename = "/dev/rmt/$tapenum"."mn";
     }
     else
     {
       $tapename = "/dev/rmt/$tapenum"."m";
     }
   }
   elsif ($SINIXN)
   {
     $tapenum = $ENV{"FVT_TAPE$tapenum"};
     if ($no_rewind)
     {
       $tapename = "/dev/ios0/rstape${tapenum}n";
     }
     else
     {
       $tapename = "/dev/ios0/rstape$tapenum";
     }
   }
   elsif ($SCO_SV)
   {
     if ($no_rewind)
     {
       $tapename = "/dev/nrStp"."$tapenum";
     }
     else
     {
```

```
        $tapename = "/dev/rStp"."$tapenum";
      }
    }
    elsif ($REAL_AIX)
    {
      if ($no_rewind)
      {
        $tapename = "/dev/rmt$tapenum".".1";
      }
      else
      {
        $tapename = "/dev/rmt$tapenum";
      }
    }
    elsif ($WINT)
    {
      $tapename = "\\\\.\\TAPE$tapenum";
    }
    else
    {
      print "Undefined operating system...cannot get tapename\n";
    }

    return ($tapename);
}


# This macro queries if the specified tape drive is physically connected
# to the system. The argument to this macro is the tape number (0-based)
sub fvt_is_tape_connected
{
   local($tape_num)=@_;
   local($tapename,$tape_opt)="";
   local($tape_exists)=0;
   $tapename = &fvt_get_tapename($tape_num,0);
   if ($REAL_AIX || $SINIXN || $SunOS)
   {
     $tape_opt = "status";
   }
   elsif ($HPUX)
   {
     $tape_opt = "rew";
   }
   system("$tapecmd $tapename $tape_opt");
   if ($? eq 0)
   {
     $tape_exists = 1;
     print "Tape Drive $tape_num is connected: $tapename\n";
   }
   else
   {
     print "Tape drive $tape_num is not connected: $tapename\n";
   }
   return($tape_exists);
}

# This macro returns the total number of tape drives connected to
# the system. This macro takes no argument
sub fvt_get_num_tape_connected
{
   local($num_of_tape)=0;
   local($MAX_TAPES) = 5;
   for ($i=0; $i < $MAX_TAPES; $i++)
   {
     if(&fvt_is_tape_connected($i))
     {
       $num_of_tape++;
     }
   }
   if ($num_of_tapes eq 0)
   {
     print("\nNOTE: no tape is connected to system\n");
   }
   return($num_of_tapes);
}

# This macro tests to see if ADSM (ADSTAR Distributed Storage
Management)
# is installed in the system.  If adsm is installed, return 1; 0 otherwise
sub fvt_is_adsm_present
{
  if ($AIX)
  {
    if (open(aFILE, "/usr/lib/libApiDS.a") || open(bFILE, "libApiDS.so"))
    {
      return(1);
    }
    else
    {
      return(0);
    }
  }
}

##External
#is_client_server
#
#  Returns 1 if this is running in a client/server environment
#

sub is_client_server
{
  local( $cs );


  if (defined $ENV{CLIENT_SERVER})
  {
     $cs = 1;
  }
  else
  {
     $cs = 0;
  }

  return $cs;

}


##External
#is_ddcs
#
#  Returns 1 if this is running in a ddcs environment
#

sub is_ddcs
{
  local( $cs );


  $cs = 0;
  if ($ENV{CS_SERVER} eq $ENV{CS_SERVER1} && ($ENV
{MVSAPPC} || $ENV{MVSTCPIP}))
  {
     $cs = 1;
  }

  return $cs;
}


##External
#is_loopback
#
#  Returns 1 if this is running in a loopback client/server environment
#

sub is_loopback
{
  local( $cs );


  $cs = 0;
  if (&is_client_server( ) && $ENV{LOOPBACK} )
  {
     $cs = 1;
  }

  return $cs;

}


##Heading
```

```
#SMP Macros


##External
#SMP variables
#
#
#  The following environment variables are set by the regression tools
#  when buckets are run in SMP mode.  They need to be set maually if
#  a bucket is to be run in SMP mode outside of regression.
#
#  FVT_SMP_DEGREE - set to the number of degrees (cpus) the bucket
should
#                  execute with
#  DB2_DEBUG_SET_NUM_CPUS - same as FVT_SMP_DEGREE but for
parallel load
#
#  In addition, the following two db2 commands must be issued:
#     db2 update dbm cfg using parallel_enable on
#     db2 update dbm cfg using max_querydegree (degree from above)
#


##External
#is_SMP
#
#  Returns 1 if the SMP degree is set to higher than 1
#

sub is_SMP
{
   local( $smp );


   if ($ENV{FVT_SMP_DEGREE})
   {
      $smp = 1;
   }
   else
   {
      $smp = 0;
   }

   return $smp;

}


##Heading
#MPP Macros


##External
#createdb
#
# macro to create a database
#
# usage:
#  createdb("dbname"[,"dbpath"][,"dboptions"]);
#
# This function should be used instead of an explicit
# call to system("db2 create..."
# Only specify a path if the bucket relies on it's
# location to operate correctly
#
# Note: The IGNORE_CLIENTSERVER environment variable will cause this
macro
#       to execute the command locally
#
# examples:
#    &createdb("testdb");
# or
#    &createdb("globaldb", "/smallfs", "ALIAS gdb");
#
# If the environment is MPP and the environment variable FVT_VARYCAT
# is set to anything then catalog node and adjustment of
IBMDEFAULTGROUP
# nodegroup are determined as follows.
#
# Layout, based on number of nodes
#
# Coord:  Coordinator node (default run location for test objects, where rbkt
command issued)
# Type:   (number of nodes -1) modulo 4
# Cat:    Catalog node
# Data:   Data placed here (part of ibmdefaultgroup)
#
#      Node     1         2         3      4      5     6     ...
#            Coord
# Type
#   2       Cat data
#   1          Data    Cat Data
#   2       Cat Data        Data    Data
#   3                 Cat         Data    Data
#   0                 Cat Data    Data    Data    Data
#   1          Data    Cat Data    Data    Data    Data   Data

sub createdb
{
   local( $dbname, $orpath, $dbopts ) = @_;
   local( $notnfsdir );
   local( @create_cmd, @node_array );
   local( $varytype, $numnodes, $temp );
   local( $currentnode, $currentnode_index, $catnode, $catnode_index );
   local( $current_hostname, $cat_hostname );

   if (&is_ddcs())
   {
     &fvt_ddcs_catalog_db($dbname,$dbopts);
     return;
   }
   elsif (&is_client_server() && !$ENV{IGNORE_CLIENTSERVER} )
   {
     &fvt_create_catalog_remote_db( "tcpip", $dbname, "", $dbopts );
     return;
   }

   if (!$orpath)
   {
      $orpath = &notnfsdir;
   }

   if ($orpath)
   {
      $notnfsdir = "on $orpath";
   }

   push( @create_cmd, "create database $dbname $notnfsdir $dbopts" );
   if ($ENV{FVT_SMP_DEGREE})
   {
     push( @create_cmd, "update db cfg for $dbname using dft_degree $ENV
{FVT_SMP_DEGREE}" );

     # temporary .... John H will assess the long term need for this

     push( @create_cmd, "update db cfg for $dbname using dbheap 4800" );
   }
   push( @create_cmd, "connect reset" );

   # set default to not vary catalog node

   $varytype = 2;

   if ($ENV{FVT_VARYCAT})
   {
      if (&is_MPP_multinode( ))
      {
          # get_cfg_nodes already called by is_MPP_multinodes
          # so environment variables are set
          $numnodes = $ENV{MPP_NODES};
          # Determine variation type
          $varytype = ($numnodes-1) % 4;
          $currentnode = &fvt_get_local_node;
          # Determine a node to use for catalog node
          if ($varytype != 2)
          {
             for ($j = 0; $j < $numnodes; $j ++)
             {

                $temp = sprintf( "MPP_NODE%4.4d", $j );
                push(@node_array,$ENV{$temp});
                # get index in the array of hosts for current node
                if ($currentnode ==  $node_array[$j])
                {
```

```
                $currentnode_index = $j;
            }
        }
        $catnode_index = ($currentnode_index+1) % $numnodes;
        $catnode = $node_array[$catnode_index];
    }
    else
    {
        $catnode = $currentnode;
    }
    #print("Environment variable FVT_VARYCAT defined.\n");
    #print("Coordinator node=$currentnode.  Catalog
node=$catnode.\n");
    if ($varytype == 0)
    {
        # take current node out of IBMDEFAULTGROUP nodegroup
        push( @create_cmd, "connect to $dbname");
        push( @create_cmd, "alter nodegroup ibmdefaultgroup drop node
($currentnode)");
        push( @create_cmd, "connect reset" );
    }
    if ($varytype == 3)
    {
        # take current node & catalog node out of IBMDEFAULTGROUP
nodegroup
        push( @create_cmd, "connect to $dbname");
        push( @create_cmd, "alter nodegroup ibmdefaultgroup drop node
($currentnode,$catnode)");
        push( @create_cmd, "connect reset" );
    }
  }
}
push( @create_cmd, "terminate" );
if ($varytype == 2)
{
    &attach( @create_cmd );
}
else
{
    &fvt_remote_db2_on_node($catnode, "", @create_cmd);
}
}


##External
#db2sampl
#
# Create the sample database.
# Can take an optional path if the default Database path is not suitable.
# The proper default path is taken for MPP setup.
# In client_server, it will create the sample db on the server and catalog it
# on the client.

sub db2sampl
{
   local($path)=@_;
   local($node) = "REMNODE";


   if(!$path)
   {
      $path = &notnfsdir();
   }

   if ($ENV{'CLIENT_SERVER'})
   {
      &remote_shell_path( $ENV{CS_SERVER}, "db2sampl $path" );
      &db2("uncatalog database sample",
          "catalog database sample at node $node",
          "terminate"
          );
   } else
   {
      &verbose_system("db2sampl $path");
   }

   if (&is_SMP( ) )
   {
     &db2start( );
     &fvt_update_db_cfg( "sample", "dft_degree $ENV
{FVT_SMP_DEGREE}" );
     &fvt_update_db_cfg( "sample", "dbheap 4800" );
   }
}


##Internal
#fvt_append_env_var_list
#
# macro to append environment variables list
#

sub fvt_append_env_var_list
{
  local( $list ) = @_;

  $list .= "CC=$ENV{CC} "                if( defined( $ENV{CC} ) );
  $list .= "C_FLAGS=$ENV{C_FLAGS} "      if( defined( $ENV{C_FLAGS}
) );
  $list .= "C_INCS=$ENV{C_INCS} "        if( defined( $ENV{C_INCS} ) );
  $list .= "C_LIBS=$ENV{C_LIBS} "        if( defined( $ENV{C_LIBS} ) );
  $list .= "CSTDLIBS=$ENV{CSTDLIBS} "    if( defined( $ENV
{CSTDLIBS} ) );
  $list .= "C_OBJS=$ENV{C_OBJS} "        if( defined( $ENV{C_OBJS} ) );
  $list .= "FC=$ENV{FC} "                if( defined( $ENV{FC} ) );
  $list .= "F_FLAGS=$ENV{F_FLAGS} "      if( defined( $ENV{F_FLAGS} )
);
  $list .= "F_INCS=$ENV{F_INCS} "        if( defined( $ENV{F_INCS} ) );
  $list .= "F_LIBS=$ENV{F_LIBS} "        if( defined( $ENV{F_LIBS} ) );
  $list .= "FSTDLIBS=$ENV{FSTDLIBS} "    if( defined( $ENV
{FSTDLIBS} ) );
  $list .= "F_OBJS=$ENV{F_OBJS} "        if( defined( $ENV{F_OBJS} ) );
  $list .= "CBLC=$ENV{CBLC} "            if( defined( $ENV{CBLC} ) );
  $list .= "CBL_FLAGS=$ENV{CBL_FLAGS} "  if( defined( $ENV
{CBL_FLAGS} ) );
  $list .= "CBL_INCS=$ENV{CBL_INCS} "    if( defined( $ENV
{CBL_INCS} ) );
  $list .= "CBL_LIBS=$ENV{CBL_LIBS} "    if( defined( $ENV
{CBL_LIBS} ) );
  $list .= "CBLSTDLIBS=$ENV{CBLSTDLIBS} " if( defined( $ENV
{CBLSTDLIBS} ) );
  $list .= "CBL_OBJS=$ENV{CBL_OBJS} "    if( defined( $ENV
{CBL_OBJS} ) );
  $list .= "COB_PREP=$ENV{COB_PREP} "    if( defined( $ENV
{COB_PREP} ) );
  $list .= "PREP_OPTS=$ENV{PREP_OPTS} "  if( defined( $ENV
{PREP_OPTS} ) );
  $list .= "BIND_OPTS=$ENV{BIND_OPTS} "  if( defined( $ENV
{BIND_OPTS} ) );
  $list .= "CNCT_OPTS=$ENV{CNCT_OPTS} "  if( defined( $ENV
{CNCT_OPTS} ) );
  $list .= "RUN_OPTS=$ENV{RUN_OPTS} "    if( defined( $ENV
{RUN_OPTS} ) );
  $list .= "TESTDIR=$ENV{TESTDIR} "      if( defined( $ENV
{TESTDIR} ) );
  $list .= "DB2OPTIONS=$ENV{DB2OPTIONS} " if( defined( $ENV
{DB2OPTIONS} ) );

  # nls environment variables supported by regression macros

  $list .= "LANG=$ENV{LANG} " if( defined( $ENV{LANG} ) );
  $list .= "LC_ALL=$ENV{LC_ALL} " if( defined( $ENV{LC_ALL} ) );
  $list .= "LC_CTYPE=$ENV{LC_CTYPE} " if( defined( $ENV
{LC_CTYPE} ) );
  $list .= "LC_MESSAGES=$ENV{LC_MESSAGES} " if( defined( $ENV
{LC_MESSAGES} ) );
  $list .= "DB2CODEPAGE=$ENV{DB2CODEPAGE} " if( defined( $ENV
{DB2CODEPAGE} ) );
  $list .= "DB2COUNTRY=$ENV{DB2COUNTRY} " if( defined( $ENV
{DB2COUNTRY} ) );

  return $list;
}


##External
#fvt_cat_to_filehandle
#
# macro to concatenate a file to another file given by filehandle.
#
# example:
#    &fvt_cat_to_filehandle( FHANDLE,"test/src/foo.c" );
#
```

```
sub fvt_cat_to_filehandle{
  local(*LOCALOUTFILE,$filename) = @_;
  local(*IN);

  if ( open(IN, "<$filename") ) {
    while (<IN>) {
      print LOCALOUTFILE $_;
    }
    close(IN);
  }
}


##External
#fvt_cleanup_autold_environment
#
# Currently only needed for mpp (autold bucket).
#
#  &fvt_cleanup_autold_environment
#

sub fvt_cleanup_autold_environment
{
   if ($AIX)
   {
      &rm ("$ENV{HOME}/.netrc");
   }
}


##External
#fvt_count_db2_threads
#
# Returns the number of currently running instances of a db2 thread
# (or process).
#
# This macro is ONLY valid for MPP platforms !!!
#
# threadname  - the name of the thread to check (ie. db2agent)
# state       - if relevant, the name of the database the thread
#               is connected to, or "idle". If this parameter is
#               empty, all threads of the requested type will be
#               counted, regardless what state they are in...
#
# Please note that this macro does little or no error checking.  Do
# not pass it data in <state> if it makes no sense to do so.
#
# Example: $running = &fvt_count_db2_threads( "db2agent", "idle" );
#            -- $running is the number of idle db2agents
#
#          $running = &fvt_count_db2_threads( "db2agntp", "mydb" );
#            -- $running is the number of db2agntp connected to mydb
#
#          $running = &fvt_count_db2_threads( "db2agent" );
#            -- $running is the number of ALL db2agents
#
#          $running = &fvt_count_db2_threads( "db2sysc" );
#            -- $running is the number of all db2sysc

sub fvt_count_db2_threads
{
  local( $process, $state ) = @_;
  local( $instances ) = 0;


  if (!&is_MPP( ) )
  {
    die "fvt_count_db2_threads is currently only supported in MPP.\n";
  }

  if ( ($OS2) || ($WINT) )
    {  die "fvt_count_db2_threads is currently only supported on UNIX.\n";  }
  elsif ($AIX)
  {
    local( $searchfor, *DATA, @process );

    $searchfor = "$process" . ($state ne "" ? " \\\($state\\\)" : "");
    open( DATA, "ps x |" );
    while( <DATA> )
    {
      chop;
      s/^[ \t\n]+//;       # Trim off leading whitespace
      @process = split( /[ \t\n]+/, $_, 5 );
      $instances += 1 if( $process[4] =~ /^$searchfor/i);   # Note: this isn't
foolproof, but should be okay...
    }
    close( DATA );
  }

  return($instances);
}


##External
#&fvt_db2nodes_entry_exists
#
# Returns TRUE ("1") if an entry having the specified values is present
#  in the 'db2nodes.cfg'.
#
# Syntax: &fvt_db2nodes_entry_exists(node number, node name, port
number<, netname>);
#
# Example: &fvt_db2nodes_entry_exists("0", "node1", "1");
# Example: &fvt_db2nodes_entry_exists("0", "node1", "1", "netname1");

sub fvt_db2nodes_entry_exists {
  local ($cfgfile) = &get_db2nodes_cfg;
  local ($matched) = 0;
  local ($nodenum, $nodename, $nodeport, $nodenetname) = @_;
  if ( ! $nodenetname ) {$numcols = 3;}
  else {$numcols = 4;}
  open (CFGFILE, $cfgfile);
  while (<CFGFILE>) {
    if ($numcols == 3) {
      if (/\s*$nodenum\s*$nodename\s*$nodeport.*$/) { $matched = 1; }
    }
    if ($numcols == 4) {
      if (/\s*$nodenum\s*$nodename\s*$nodeport\s*$nodenetname\s*.*$/)
{ $matched = 1; }
    }
  }
  close (CFGFILE);
  return $matched;
}


##External
#fvt_db2nodes_file_delete
#
# Deletes the 'db2nodes.cfg' file
#
# Example: &fvt_db2nodes_file_delete();

sub fvt_db2nodes_file_delete {
  local ($cfgfile) = &get_db2nodes_cfg;
  local ($retcode);
  if (! -f $cfgfile) { $retcode = -1}
  else {
    &rm($cfgfile);
    $retcode = 0;
  }
  if ($retcode == -1) {printf("\$cfgfile\' does not exist.\n")}
  return $retcode;
}


##External
#fvt_db2nodes_last_line_del_valid
#
# macro to delete last line from db2nodes.cfg, and save
# the node number, node name, port number, net name values from that line.
#
# usage:
#      &fvt_db2nodes_last_line_del_valid( );
#

sub fvt_db2nodes_last_line_del_valid
{
  local( $cfgname, $j, $numnodes );
  local( @input );
  local( $nnumlast, $nnamelast, $mlnlast, $routelast );


  $cfgname = &get_db2nodes_cfg( );
```

```
  open( NODES, $cfgname );
  @input = <NODES>;
  close( NODES );

  $numnodes = scalar(@input);

  if ($numnodes < 2 )
  {
    printf( "Node configuration file only has $numnodes nodes,\n" );
    printf( "not enough nodes to process.\n" );
    return -1;
  }
  &rm($cfgname);

  open( NODES, ">$cfgname" );

  for ($j = 0; $j < ($numnodes-1); $j ++)
  {

#     &debug_printf( "Writing line $j to node configuration file: $input[$j]" );
    print NODES "$input[$j]";
  }
  $last = $numnodes-1;
  ( $nnumlast, $nnamelast, $mlnlast, $routelast ) = split( /\s+/, $input[$last] );
  $ENV{MPP_SAVED_NODE} = $nnumlast;
  $ENV{MPP_SAVED_NAME} = $nnamelast;
  $ENV{MPP_SAVED_PORT} = $mlnlast;
  $ENV{MPP_SAVED_ROUTE} = $routelast;
  close( NODES );
  return 0;

}


##External
#fvt_db2nodes_last_node_replace_valid
#
# macro to replace the nodename, port number, net name values
# of last node in db2nodes.cfg with the values saved
# from fvt_db2nodes_last_line_del_valid.  That node keeps same
# node number.  This macro must be invoked in same program
# that fvt_db2nodes_last_line_del_valid is invoked in.
#
# usage:
#       &fvt_db2nodes_last_node_replace_valid( );
#

sub fvt_db2nodes_last_node_replace_valid
{
  local( $cfgname, $j, $numnodes );
  local( @input );
  local( $nnumlast, $nnamelast, $mlnlast, $routelast );
  local( $nodename, $mln, $route );

  if ($ENV{MPP_SAVED_NAME} eq "" )
  {
    printf( "There is no saved data to use.\n" );
    return -1;
  }

  $cfgname = &get_db2nodes_cfg( );

  open( NODES, $cfgname );
  @input = <NODES>;
  close( NODES );

  $numnodes = scalar(@input);

  if ($numnodes < 1 )
  {
    printf( "Node configuration file only has $numnodes nodes,\n" );
    printf( "not enough nodes to process.\n" );
    return -1;
  }
  &rm($cfgname);

  open( NODES, ">$cfgname" );

  for ($j = 0; $j < ($numnodes-1); $j ++)
  {
#     &debug_printf( "Writing line $j to node configuration file: $input[$j]" );
    print NODES "$input[$j]";
  }
  $last = $numnodes-1;
  ( $nnumlast, $nnamelast, $mlnlast, $routelast ) = split( /\s+/, $input[$last] );
  $nodename = $ENV{MPP_SAVED_NAME};
  $mln = $ENV{MPP_SAVED_PORT};
  $route = $ENV{MPP_SAVED_ROUTE};
  print NODES "$nnumlast $nodename $mln $route\n";
  close( NODES );
  return 0;

}


##External
#fvt_db2nodes_manipulate
#
# A tool to do all sorts of useful stuff to and with the db2nodes.cfg file.
# This command will accept multiple instructions, which will be executed in
# order on the db2nodes.cfg file.  Each instruction and its arguments should
# be passed as one argument to this function.
#
# This command ALWAYSreturns a list. The first element in this list will
# always be the number of nodes in the current db2nodes.cfg.  If none of the
# ENUM* instructions are given, the returned list will have only one element.
#
#
*************************************************************
******
# **** IMPORTANT NOTE  ****  IMPORTANT NOTE  ****  IMPORTANT
NOTE  ****
# ****  DO NOT use this macro with any other db2nodes.cfg macros   ****
# ****  UNLESS you keep all calls to it together, and finish each  ****
# ****  group with a CLEANUP.                                      ****
#
*************************************************************
******
#
# All examples that follow build on the previous examples, and will execute
# using the following original db2nodes.cfg:
# 3  earth 0 earth.torolab.ibm.com
# 4  wind  0
# 9  fire  0
# 10 earth 1 earth.torolab.ibm.com
# 13 wind  1
# 25 wind  2
# 33 earth 2 earth.torolab.ibm.com
# 34 fire  1
#
# Supported commands:
#    cleanup   - should be passed to this function by itself (don't invoke it
#                in a list of commands).  This instruction will cause the
#                macro to clean up after itself, restoring the db2nodes.cfg
#                state to what it was before the first call to this macro.
#                YOU SHOULD EXECUTE THIS INSTRUCTION before exiting
your bucket.
#
#                The cleanup command can also be used in the special case where
#                your bucket needs to be run with node numbers consecutive
#                from an arbitrary point (usually 0).  In this case, the phrase
#                "relabel x" should be used to cause cleanup to relabel the
#                ***ORIGINAL*** db2nodes.cfg file with node numbers starting at
#                "x".
#                ie:
#                   "cleanup relabel 0"
#                will result in the *ORIGINAL* db2nodes.cfg file being
#                recreated with node numbers starting at 0.
#
#                In general, you should NOT need to use the relabel
#                functionality of cleanup.
#
#    length    - Used to change the number of nodes listed in db2nodes.cfg.
#                This instruction is not capable of lengthening the node list
#                beyond the length it was initially set to.  This instruction
#                takes one argument - the number of nodes you want to use.
#                ie:
#                   "length 3"
#                will result in:
#                   3  earth 0
#                   4  wind  0
#                   9  fire  0
```

```
#
#   drop      - Used to drop nodes from the working db2nodes.cfg.  This
#               instruction takes one argument - a range of nodes you want
#               dropped.  NOTE: the range refers to offsets into the
#               original db2nodes.cfg, not to node numbers.
#               ie:
#                 "drop 0,3"
#               will result in (note line 3 is already missing):
#                 4  wind  0
#                 9  fire  0
#
#   add       - Used to add nodes to the working db2nodes.cfg.  The nodes
#               are added from the original db2nodes.cfg, and are inserted
#               in the same ordering they would originally have used.
#               This instruction takes one argument - a range of nodes you
#               want added back.  NOTE: the range refers to offsets into
#               the original db2nodes.cfg, not to node numbers.
#               ie:
#                 "add 4-6,7"
#               will result in:
#                 4  wind  0
#                 9  fire  0
#                 13 wind  1
#                 25 wind  2
#                 33 earth 2
#                 34 fire  1
#
#   enum      - Used to return a list of the current node numbers.
#               ie:
#                 "enum"
#               will result in the following returned list (see above for
#               an explanation of the first list element):
#                 ( 6, 4, 9, 13, 25, 33, 34 )
#
#   enumnames - Used to return a list of current node host names.
#               ie:
#                 "enumnames"
#               will result in the following returned list (see above for
#               an explanation of the first list element):
#                 ( 6, 'wind', 'fire', 'wind', 'wind', 'earth', 'fire' )
#
#   enummln   - Used to return a list of db2nodes.cfg offsets, grouped by
#               host name.  The list will take the form:
#                 (hostname, number of nodes, [offsets] ... )
#               ie:
#                 "enummln"
#               will result in the following returned list (see above for
#               an explanation of the first list element):
#                 ( 6, 'wind', 3, 1, 4, 5, 'fire', 2, 2, 7, 'earth', 1, 6 )
#
#   enuminterleave - Used to return a list of nodes number and host names.
#               ie:
#                 "enuminterleave"
#               will result in the following returned list (see above for
#               an explanation of the first list element):
#                 ( 6, 4, 'wind', 9, 'fire', 13, 'wind', 25, 'wind', 33,
#                   'earth', 34, 'fire' )
#
#   enumall   - Used to return a list of all information in db2nodes.cfg.
#               ie:
#                 "enumall"
#               will results in the following returned list (see above for
#               an explanation of the first list element):
#                 ( 6, 4, 'wind',  0, '',
#                      9, 'fire',  0, '',
#                     13, 'wind',  1, '',
#                     25, 'wind',  2, '',
#                     33, 'earth', 2, 'earth.torolab.ibm.com',
#                     34, 'fire',  1, '', )
#
# Usage:
#   ($numnodes, @nodelist) = &fvt_db2nodes_manipulate( "length 5",
#                                                      "drop 1-2,4",
#                                                      "add 7",
#                                                      "enum" );
#   # Your bucket code goes here...
#   &fvt_db2nodes_manipulate( "cleanup" );
#

sub fvt_db2nodes_manipulate
{
  local( # Initialized
      $TRUE, $FALSE,
      $fileShadowed, $ShadowFile, $ConfigFile, $ShadowBase, $retVal,
      @retNodeList, $cfgChanged, $",

      # Uninitialized
      $arg, $instruction, @arglist, $counter, $done,
      @range, @range2, $line,
      *FILE, @originalCfg, @currentCfg, $i, $j, @cfgLine, @cfgLine2,
      %handled, @innerCfgLine, $countAt, $empty, $garbage
    );

  $TRUE  = 1;
  $FALSE = 0;
  $" = ' ';

  if( $ENV{TESTDIR} )
    { $ShadowBase = $ENV{TESTDIR}; }
  elsif( $ENV{HOME} )
    { $ShadowBase = $ENV{HOME}; }
  else
    { die "fvt_db2nodes_manipulate: \$ENV{TESTDIR} or \$ENV{HOME}
must be set!\n"; }
  $ShadowBase =~ s</$><>;

  $ConfigFile = &get_db2nodes_cfg;
  $ShadowFile = $ConfigFile;
  if( $ConfigFile =~ /.*\/.*/ )
    { $ShadowFile =~ s<.*/(.*)$><$ShadowBase/_$1>; }
  else
    { $ShadowFile = "$ShadowBase/_$ShadowFile"; }
  $fileShadowed = $FALSE;

  $retVal = 0;
  @retNodeList = ();
  $cfgChanged = $FALSE;

  #*************************************************************
*****
  # Matches CLEANUP - CLEANUP and exit.  If command is
  # CLEANUP RELABEL x, then renumber the  ***ORIGINAL***
db2nodes.cfg
  # consecutively from x.
  if( $_[0] =~ /^CLEANUP/i )
  {
    if( -e $ShadowFile )
    {
      chmod 0666, ($ConfigFile, $ShadowFile);
      &cp( $ShadowFile, $ConfigFile );
      &rm( $ShadowFile );
    }

    open( FILE, $ConfigFile );
    chop(@originalCfg = <FILE>);
    close( FILE );

    # Relabel code.  This whole function needs to be re-written, methinks... CP
    ($instruction, @arglist) = split( /[ \n\t]+/, $_[0] );
    if( $arglist[0] =~ /^RELABEL$/i )
    {
      local( $catnode, $catline, @working );

      $catnode = &fvt_get_catalog_node;

      # Build a list of <portno hostname netname>
      foreach $line (@originalCfg)
      {
        @cfgLine = split( /[ \t]+/, $line );

        if( $cfgLine[0] == $catnode )
          { $catline = sprintf( "%-6d %-15s %5d %s", 0+$arglist[1], @cfgLine
[1..3] ); }
        else
          { @working[++$#working] = sprintf( "%05d %40s %s", $cfgLine[2],
$cfgLine[1], $cfgLine[3] ); }
      }
      @working = sort @working;

      # Assemble the new list
      @originalCfg = ( $catline );
      $i = 1;
      foreach $line (@working)
      {
```

```
        @cfgLine = split( /[ ]+/, $line );
        @originalCfg[scalar(@originalCfg)] = sprintf( "%-6d %-15s %5d %s",
($i++ + $arglist[1]), $cfgLine[1], $cfgLine[0], $cfgLine[2] );
      }

      open( FILE, ">$ConfigFile" );
      $" = "\n";
      print FILE "@originalCfg\n";
      $" = " ";
      close( FILE );
    }

    return scalar( @originalCfg );
  }
  #*************************************************************
*****

  # Make a backup of db2nodes.cfg
  chmod 0666, ($ConfigFile);
  if( !(-e $ShadowFile) )
    {  &cp( $ConfigFile, $ShadowFile );  }

  # Read in both the original and the current node lists.
  open( FILE, "$ShadowFile" );
  @originalCfg = <FILE>;
  chop @originalCfg;
  close( FILE );
  open( FILE, "$ConfigFile" );
  @currentCfg = <FILE>;
  chop @currentCfg;
  close( FILE );

  $counter = 0;
  outer: while( $counter < scalar( @_ ))
  {
    $arg = $_[$counter];
    ($instruction, @arglist) = split( /[ \n\t]+/, $arg );
    $instruction =~ /(^LENGTH$)|(^DROP$|^ADD$|^ENUMMLN$)|
(^ENUM$|^ENUMNAMES$|^ENUMINTERLEAVE$|^ENUMALL$)/i;
    next;
  }
  continue
  {
    $counter++;
    $done = $FALSE;


  #*************************************************************
*****
  # Matches LENGTH
    LENGTH: {
      next if( $done || ( $1 eq "" ));

      # LENGTH <numnodes> [startingat]
      if( $arglist[0] > scalar( @originalCfg ))
        {   $retVal = -1;  }
      else
      {
        $#currentCfg = ();
        for( $i = 0; $i < $arglist[0]; $i++ )
          {  $currentCfg[$i] = $originalCfg[$i];  }
      }
      $cfgChanged = $TRUE;
      $done = $TRUE;
    }
  #*************************************************************
*****


  #*************************************************************
*****
  # Matches any functions that need to pair up the current
  # and original files: DROP or ADD
    MATCH: {
      next if( $done || ($2 eq "" ));

      # Match the current nodes to the original nodes
      for( $i = $#currentCfg; $i>=0; $i-- )
      {
        $_ = $currentCfg[$i];
        @cfgLine = split;
        for( $j = $#originalCfg; $j>=0; $j-- )
        {
          $_ = $originalCfg[$j];
          @cfgLine2 = split;

          if( ($cfgLine2[1] eq $cfgLine[1]) && ($cfgLine2[2] == $cfgLine[2])
&& ($i != $j))   # Match on name and port, line not already matched...
          {
            $currentCfg[$j] = $currentCfg[$i];
            $currentCfg[$i] = "";
            last;
          }
        }
      }

      DROP: {
        next if( $instruction !~ /^DROP$/i );

        # DROP <range of original indices>
        @range = &fvt_interpret_range( "@arglist" );
        foreach $line (@range)
        {
          $currentCfg[$line] = "";
        }
        $cfgChanged = $TRUE;
      }
      ADD: {
        next if( $instruction !~ /^ADD$/i );

        # ADD <range of original indices>
        @range = &fvt_interpret_range( "@arglist" );

        for( $i=0; $i<scalar(@range); $i++ )
        {
          $line = $range[$i];
          $currentCfg[$line] = $originalCfg[$line];
        }
        $cfgChanged = $TRUE;
      }
      ENUMMLN: {
        next if( $instruction !~ /^ENUMMLN$/i );

        %handled = ();
        print "In ENUMMLN\n";
        for( $i=0; $i<scalar(@currentCfg); $i++ )
        {
          $_ = $currentCfg[$i];
          @cfgLine = split;

          if( ($cfgLine[1] ne "") && ($handled{ $cfgLine[1] } eq "") )
          {
            $handled{ $cfgLine[1] } = "done";
            push( @retNodeList, $cfgLine[1] );
            push( @retNodeList, 0 );
            $countAt = $#retNodeList;
            for( $j = $i; $j<scalar(@currentCfg); $j++ )
            {
              $_ = $currentCfg[$j];
              @innerCfgLine = split;

              if( $innerCfgLine[1] eq $cfgLine[1] )
              {
                push( @retNodeList, $j );
                $retNodeList[$countAt]++;
              }
            }
          }
        }
      }

      # Resort the config
      $empty = -1;
      for( $i=0; $i<=$#currentCfg; $i = ($i < $empty ? $empty + 1 : $i + 1 ) )
      {
        if( $currentCfg[$i] eq "" )
          {  $empty = $i if( $empty == -1 );  }
        else
        {
          if( $empty != -1 )
          {
            $j = $i;
            while( ($j<=$#currentCfg) && ($currentCfg[$j] ne "")  )
            {
```

```
                $currentCfg[$empty++] = $currentCfg[$j];
                $currentCfg[$j] = "";
                $j++;
              }
            }
          }
        }
        if( $empty != -1 )
          {  $#currentCfg = $empty - 1;  }

        $done = $TRUE;
      }
  #*************************************************************
*****


  #*************************************************************
*****
  # Matches ENUM, ENUMNAMES, ENUMINTERLEAVE, and ENUMALL
    ENUM: {
      next if( $done || ($3 eq "" ));
      foreach $line (@currentCfg)
      {
        $_ = $line;
        @cfgLine = split;

        if( $instruction =~ /(^ENUM$|^ENUMINTERLEAVE$|
^ENUMALL$)/i )
          {  push( @retNodeList, $cfgLine[0] );  }
        if( $instruction =~ /(^ENUMNAMES$|^ENUMINTERLEAVE$|
^ENUMALL$)/i )
          {  push( @retNodeList, $cfgLine[1] );  }
        if( $instruction =~ /(^ENUMALL$)/i )
        {
          if( @cfgLine[2] ne "" )
            {  push( @retNodeList, $cfgLine[2] );  }
          else
            {  push( @retNodeList, 0 );  }

          if( @cfgLine[3] ne "" )
            {  push( @retNodeList, $cfgLine[3] );  }
          else
            {  push( @retNodeList, "" );  }
        }
      }
      $done = $TRUE;
    }
  #*************************************************************
*****

  }
  if( $cfgChanged )
  {
    open( FILE, ">$ConfigFile" );
    foreach $line (@currentCfg)
    {
      print FILE "$line\n";
    }
    close( FILE );
  }

  $retVal = scalar( @currentCfg );

  return ($retVal, @retNodeList);
}


##External
#fvt_db2nodes_random_one_valid
#
# For regression run of test bucket, the configuration file db2nodes.cfg is
# set up with first node as the node where the test bucket is running from.
# This macro alters the db2nodes.cfg, to change the first node to a
# new position in the node order of db2nodes.cfg.  The new position is
# randomly chosen.  The node number sequence is maintained to keep
# db2nodes.cfg valid, only the hostname, logical-port, netname for
# the node are moved.
#
# usage:
#      &fvt_db2nodes_random_one_valid( );
#
sub fvt_db2nodes_random_one_valid
{
  local( $cfgname, $j, $numnodes );
  local( @input );
  local( $new_position );
  local( $nnum1, $nname1, $mln1, $route1 );
  local( $nnum2, $nname2, $mln2, $route2 );

  $cfgname = &get_db2nodes_cfg( );

  open( NODES, $cfgname );
  @input = <NODES>;
  close( NODES );

  $numnodes = scalar(@input);
  if ($numnodes < 1 )
  {
    # There is nothing to do.  Just end.
    return -1;
  }

  $new_position = int(rand($numnodes));
#  &debug_printf( "The randomn value generated is : $new_position\n" );
  if ($new_position == 0) {
#  &debug_printf( "The randomn number chosen keeps file as it is.  Nothing to
do.\n" );
    return 0;
  }

  &rm($cfgname);

  open( NODES, ">$cfgname" );

  ( $nnum1, $nname1, $mln1, $route1 ) = split( /\s+/, $input[0] );
  ( $nnum2, $nname2, $mln2, $route2 ) = split( /\s+/, $input[$new_position] );
  for ($j = 0; $j < $numnodes; $j ++)
  {

    if ($j == 0) {
      print NODES "$nnum1 $nname2 $mln2 $route2\n";
    } elsif ($j == $new_position) {
      print NODES "$nnum2 $nname1 $mln1 $route1\n";
    } else {
#       &debug_printf( "Writing line $j to node configuration file: $input[$j]
\n" );
      print NODES "$input[$j]";
    }

  }

  close( NODES );
  return 0;
}


##External
#fvt_db2nodes_restore
#
# macro to restore node configuration from a node configuration saved
# with macro fvt_db2nodes_save.
#
# example:
#      &fvt_db2nodes_restore( "setup1" );
#
sub fvt_db2nodes_restore
{
  local( $name ) = @_;

  $cfgname = &get_db2nodes_cfg( );
  $cfgtemp = "$cfgname.$name";

  &cp( $cfgtemp, $cfgname );
  if (!-f( $cfgname ) )
  {
    printf ("Saved configuration file $cfgname could not be restored\n");
    return -1;
  }
  return 0;
}


##External
```

```
#fvt_db2nodes_rev_lines_invalid
#
# macro to re-order the lines in db2nodes.cfg in reverse order.
# note: produces invalid db2nodes.cfg file
#
# usage:
#       &fvt_db2nodes_rev_lines_invalid( );
#
#

sub fvt_db2nodes_rev_lines_invalid
{
  local( $cfgname ) = &get_db2nodes_cfg( );
  local( $numnodes, $j );
  local( @input );

  if (!-f($cfgname) )
  {
    printf ("can not open cfg file $cfgname\n");
    return -1;
  }

  open( NODES, $cfgname );
  @input = <NODES>;
  close(NODES);

  $numnodes = scalar(@input);
  &rm($cfgname);


  open( NODES, ">$cfgname" );
  for ($j = ($numnodes-1); $j >= 0; $j --)
  {
#     &debug_printf( "Change line $j to :$input[$j]" );
     print NODES "$input[$j]";
  }
  close( NODES );
  return 0;

}


##External
#fvt_db2nodes_save
#
# macro to save current nodes configuration, referencing
# this configuration by name passed.  The saved configuration
# can be restored with macro fvt_db2nodes_restore.
#
# example:
#       &fvt_db2nodes_save( "setup1" );
#
sub fvt_db2nodes_save
{
  local( $name ) = @_;

  $cfgname = &get_db2nodes_cfg( );
  $cfgtemp = "$cfgname.$name";

  &cp( $cfgname, $cfgtemp );
  if (!-f( $cfgtemp ) )
  {
    printf ("Saved configuration file $cfgtemp could not be created\n");
    return -1;
  }
  return 0;
}


##External
#fvt_db2nodes_swap_last2_valid
#
# macro to alter db2nodes.cfg by swapping the hostname, port and
# netname between the last two lines.  Note that node numbers in
# lines is not changed.
#
# usage:
#       &fvt_db2nodes_swap_last2_valid( );
#

sub fvt_db2nodes_swap_last2_valid
{
  local( $cfgname, $j, $numnodes );
  local( @input );
  local( $nnum2last, $nname2last, $mln2last, $route2last );
  local( $nnumlast, $nnamelast, $mlnlast, $routelast );


  $cfgname = &get_db2nodes_cfg( );

  open( NODES, $cfgname );
  @input = <NODES>;
  close( NODES );

  $numnodes = scalar(@input);

  if ($numnodes < 2 )
  {
    printf( "Node configuration file only has $numnodes nodes,\n" );
    printf( "not enough nodes to process.\n" );
    return -1;
  }
  &rm($cfgname);

  open( NODES, ">$cfgname" );

  for ($j = 0; $j < ($numnodes-2); $j ++)
  {

#     &debug_printf( "Writing line $j to node configuration file: $input[$j]" );
     print NODES "$input[$j]";
  }
  $secondlast = $numnodes-2;
  $last = $numnodes-1;
  ( $nnum2last, $nname2last, $mln2last, $route2last ) = split( /\s+/, $input
[$secondlast] );
  ( $nnumlast, $nnamelast, $mlnlast, $routelast ) = split( /\s+/, $input[$last] );
#  &debug_printf( "change 2nd to last line to $nnum2last $nnamelast $mlnlast
$routelast" );
  print NODES "$nnum2last $nnamelast $mlnlast $routelast\n";
#  &debug_printf( "change last line to $nnumlast $nname2last $mln2last
$route2last" );
  print NODES "$nnumlast $nname2last $mln2last $route2last\n";

  close( NODES );
  return 0;

}


##External
#fvt_file_to_array
#
# macro to read in a file into an array, the macro returns the array.
#
# example:
#    @myarray = &fvt_file_to_array( "test/src/foo.c" );
#

sub fvt_file_to_array{
  local($filename) = @_;
  local(@content) = ();
  local(*IN);

  if ( open(IN, "<$filename") ) {
    @content = <IN>;
    close(IN);
  }
  @content
}


##External
#fvt_find_deadlock_detector
# fvt_find_deadlock_detector
#
# Returns the number of instances fo the requested deadlock detector
# that are currently running.  Takes only one parameter, that being
# either "global" or "local", "global" is the default.
#
sub fvt_find_deadlock_detector
{
  local( $type ) = @_;
  local( $count, $process );
```

```
  $process = "db2glock";
  if ($type eq "local")
  {
    $process = "db2dlock";
  }

  if ($AIX)
  {
    chop( $count = `ps -fu $ENV{DB2INSTANCE} | grep $process | grep -v
grep | wc -l` );
  }

  return( $count );

}


##External
#fvt_for_each_node
#
# Executes a specified command on each node in 'db2nodes.cfg', using
#  the &remote_shell_path() macro.
#
#   Example: &fvt_for_each_node("ls");
#

sub fvt_for_each_node
{
  local ($command) = shift;
  local ($DB2PATH, $node, $nodenum, $junk);

  &get_sysval;
  $DB2PATH = $sysval{sqllibdir};

  open (NODEFILE, "$DB2PATH/db2nodes.cfg");
  while (<NODEFILE>) {
    $node = $_;
    chop $node;
    if ( (substr($node, 0, 1) ne "*") && ($node ne "") ) {
      ($nodenum, $node, $junk) = split(/\s+/, $_);
      &remote_shell_path($node, "$command");
    }
  }
  close NODEFILE;

}


##External
#fvt_get_catalog_node
# fvt_get_catalog_node( )
#
# Returns the node number of the catalog node assuming that the createdb
# macro is used to create the database.  It can be used before or after
# the createdb is issued (does not require connection to database).
#

sub fvt_get_catalog_node
{
   local( @node_array );
   local( $varytype, $numnodes, $temp );
   local( $currentnode, $currentnode_index, $catnode, $catnode_index );

   # Start by assuming catalog node is current node
   if (&is_MPP( ))
   {
    $currentnode = &fvt_get_local_node;
    $catnode = $currentnode;
   }
   else
   {
    $catnode = 0;
   }
   # Calculate catalog node using the same algortihm as createdb macro
   # if the FVT_VARYCAT environment variable is set and there are
   # multiple nodes
   if ($ENV{FVT_VARYCAT})
   {
      if (&is_MPP_multinode( ))
      {
         # get_cfg_nodes already called by is_MPP_multinodes
         # so environment variables are set
         $numnodes = $ENV{MPP_NODES};
         # Determine variation type
         $varytype = ($numnodes-1) % 4;
         # Determine a node to use for catalog node
         if ($varytype != 2)
         {
            for ($j = 0; $j < $numnodes; $j ++)
            {
               $temp = sprintf( "MPP_NODE%4.4d", $j );
               push(@node_array,$ENV{$temp});
               # get index in the array of hosts for current node
               if ($currentnode ==  $node_array[$j])
               {
                  $currentnode_index = $j;
               }
            }
            $catnode_index = ($currentnode_index+1) % $numnodes;
            $catnode = $node_array[$catnode_index];
         }
      }
   }
 return( $catnode );
}


##External
#fvt_get_local_node
# fvt_get_local_node( )
#
# Returns the local node number, taking into account the DB2NODE
# environment variable if it is set.
#
sub fvt_get_local_node
{

  return( &fvt_name_to_node( &get_hostname( ) ) );

}


##Internal
#fvt_list_ipcs
# fvt_list_ipcs
#
# Creates an array called '@ipcs', which contains the output from a
# 'ipcs | grep $USER | sort' command.
#
# Will only run on Unix systems.  Returns '-1' on non-Unix platforms.
# Returns '0' if successful on Unix platforms.
#
sub fvt_list_ipcs
{
  local ($ret_code) = -1;
  local ($line);

  if ($AIX) {
    open(IN, 'ipcs | grep $USER | sort |');
    while (<IN>) {
      chop( $line = $_ );
      push( @ipcs, $line );
      $ret_code = 0;
      }
    close(IN);
    }
  return ($ret_code);
}


##External
#fvt_name_to_node
# fvt_name_to_node( $host_name )
#
# Translates the host name to its corresponding node number according to
# db2nodes.cfg
# Eg: $node_number = &name_to_node( 'hawk' );
#
# Note if a host has multiple logical nodes, the multiple logical
# with port 0 is returned unless DB2NODE is specified - in which case
# that value is returned.
#
```

```
sub fvt_name_to_node
{
  local($name) = shift;
  local(*db2_cfg, $cfgfile, @hostNodes, $node, $garbage);

  $cfgfile = &get_db2nodes_cfg( );
  open(db2_cfg, "<$cfgfile") || return -1;
  while( <db2_cfg> )
  {
    chop; split;
    if ( $_[1] =~ /^$name\s*/ )
    {
      push( @hostNodes, sprintf( "%06d:%d", $_[2], $_[0] ));
    }
  }
  close(db2_cfg);
  @hostNodes = sort @hostNodes;

  if( scalar(@hostNodes) == 0 )
    {  $node = -1;  }
  elsif( scalar(@hostNodes) == 1 || (scalar(@hostNodes)>1 && (($name ne
&get_hostname) || ($ENV{DB2NODE} eq "")) ))
    {  ($garbage, $node) = split( /:/, $hostNodes[0] );  }
  else
    {  $node = $ENV{DB2NODE};  }

  return $node;
}


##Internal
#fvt_node_configure
#
# Do node configuration and necessary cleanning on a MPP slave
#
# It takes 3 arguments: nodenum, logical_nodenum, random_nodes
#

sub fvt_node_configure
{
  local($nodenum, $logical_nodenum, $random_nodes) = @_;
  local($result);
  $result = 1;

  local(@BadHostF) = ( '/net/hawk/pdbregr/badhosts' , "$ENV{'HOME'}/
badhosts" );
  foreach (@BadHostF)
  {
      local(*bh);
      open(bh,"<$_");
      while (<bh>)
      {
          chop; split;
          push(@badHosts,@_);
      }
      close(bh);
  }

  $fn_cmd = "fvt_fn -d regress -m $nodenum ";
  if ($random_nodes)
  {
    $fn_cmd .= "-g ";
  }
  if ($logical_nodenum)
  {
    if ($logical_nodenum == -1)
    {
      # wants no logical nodes!

      $fn_cmd .= "-l ";
    }
    else
    {
      # number of logical nodes is specified
      # turn this around, and tell fvt_fn that it can only have the number of
      # physical nodes that is the difference between the total # of nodes
      # and the number of logical nodes needed

      $logical_nodenum = $nodenum - $logical_nodenum - 1;
      $fn_cmd .= "-p $logical_nodenum ";
    }
  }
  $fn_cmd .= "-x " . join(' ',@badHosts) if (@badHosts);

  &debug_printf( $fn_cmd );
  $fn_result = &quiet_system_return( "$fn_cmd" );
  if ($fn_result =~ /Not enough physical nodes/)
  {
    # fn was unable to find enough nodes to run the bucket - terminate the run

    printf( "not enough nodes - fn output:\n$fn_result" );
    $result = 0;
  }

  &verbose_system( "db2_kill" );
  $db2nodesfile = "$ENV{HOME}/sqllib/db2nodes.cfg";
  &cp ($db2nodesfile, "${db2nodesfile}.fn.orig");

  return $result;
}


##External
#fvt_node_to_name
# fvt_node_to_name( $node_number );
#
#  Translates the nodenumber to the corresponding hostname
#  according to db2nodes.cfg
#  Eg: $host_name = &fvt_node_to_name( 0 );

sub fvt_node_to_name
{
  local($node) = shift;
  local(*db2_cfg, $cfgfile);


  $cfgfile = &get_db2nodes_cfg( );
  open(db2_cfg, "<$cfgfile") || return '';
  while( <db2_cfg> )
  {
    chop; split;
    if ( $_[0] eq $node )
    {
      close(db2_cfg);
      return $_[1];
    }
  }
  close(db2_cfg);
  return '';
}


##External
#fvt_remote_chmod
#
# macro to execute a chmod operation on a remote host
#
# usage:
#       &fvt_remote_chmod( "hawk", 0777, "source_file" );
#

sub fvt_remote_chmod
{
  local( $host, $mode, $source ) = @_;


  &remote_shell_path( $host, "fvt_chmod $mode $source" );

}


##External
#fvt_remote_cp
#
# macro to execute a copy operation on a remote host
#
# usage:
#       &fvt_remote_cp( "hawk", "source_file", "dest_file" );
#

sub fvt_remote_cp
{
  local( $host, $source, $dest ) = @_;
```

```
  &remote_shell_path( $host, "fvt_cp $source $dest" );

}


##External
#fvt_remote_db2_on_node
#
# This macro enables DB2 commands to be issued in a standard fashion
without
# having to know platform specific details related to system call syntax.
#
# syntax:  &fvt_remote_db2_on_node( hostname|nodenum, env,
[-<db2option>,] command1 [,command2] [,command3] ... );
#
# PLEASE NOTE THE ADDITION of the env parameter.  This string should
contain
# any extra enviroment variable sets you need.  For instance:
#    "DB2VAR=VALUE ANOTHERVAR=ANOTHERVALUE"
# If you do not wish to set up any extra variables, pass the empty string.
# Also, if the variable TESTDIR is set, the remote command will be run from
# that directory.
#
# This macro can be passed one or more options that will be passed on
# to DB2.  (ex. +v, +c, ...)  The default options that are sent at
# command invokation are +p, +t, and -v.
#
# Multiple DB2 commands can be given as parameters to this macro.  The
macro
# creates a temporary script file out of these parameters, which is executed by
# DB2.
#
# Example: &fvt_remote_db2_on_node( 1, "MYVAR=16 DB2VAR=on", "-z
outfile","+v",
#              "connect to mydb",
#              "list tables",
#              "select * from mytbl",
#              "connect reset");
#

sub fvt_remote_db2_on_node
{
  local( $tonode ) = shift;
  local( $env ) = shift;

  &fvt_remote_shell_on_node( $tonode, $env, "db2", @_ );

  return;
}


##External
#fvt_remote_db2_on_node_async
#
# Remote DB2 On Node Asynchronous:
#   does exactly what fvt_remote_db2_on_node does, except that it does not
#   wait for the remote db2 to finish before returning.
#
# See &fvt_remote_db2_on_node for details.

sub fvt_remote_db2_on_node_async
{
  local( $tonode ) = shift;
  local( $env ) = shift;

  &fvt_remote_shell_on_node_async( $tonode, $env, "db2", @_ );

  return;
}


##External
#fvt_remote_mkdir
#
# macro to execute a mkdir operation on a remote host
#
# usage:
#       &fvt_remote_chmod( "hawk", "directory" );
#

sub fvt_remote_mkdir
{
  local( $host, $dir ) = @_;


  &remote_shell_path( $host, "fvt_mkdir $dir" );

}


##External
#fvt_remote_mv
#
# macro to execute a move operation on a remote host
#
# usage:
#       &fvt_remote_mv( "hawk", "source_file", "dest_file" );
#

sub fvt_remote_mv
{
  local( $host, $source, $dest ) = @_;


  &remote_shell_path( $host, "fvt_mv $source $dest" );

}


##External
#fvt_remote_rm
#
# macro to execute a remove operation on a remote host
#
# usage:
#       &fvt_remote_rm( "hawk", "dest_file" );
#

sub fvt_remote_rm
{
  local( $host, $file ) = @_;


  &remote_shell_path( $host, "fvt_rm $file" );

}

##External
#fvt_remote_shell_on_node
#
# Remote Shell On Node:
#   run a command on a remote node (the function will automatically set up
#   the DB2NODE environment variable).  It can be passed a host name or a
#   node number.  If one of the environment variables is TESTDIR, the remote
#   command will be run in that directory.
#
# &fvt_remote_shell_on_node( &lt;hostname|node_num&gt;,
#                        string of &lt;additional_ENV_variable&gt;=&lt;its
value&gt;
#                        &lt;command&gt; );
#
# example:
#    &fvt_remote_shell_on_node( "hawk",
#                          "MYVAR=10 TESTDIR=/some/directory"
#                          "fvt_tail -$MYVAR myfile" );
#

sub fvt_remote_shell_on_node # ( $hostname_nodenum, $env, $command,
@script )
{
  local( $retVal, $tohost, $datafile ) = &fvt_remote_shell_on_node_prep
( @_ );
  if( $retVal == -1 )
    {  return -1;  }

  # Run the remote control program
  if( &get_hostname eq $tohost )
    {  &quiet_system( "fvt_command_on_node $datafile" );  }
  else
    {  &remote_shell_path( $tohost, "fvt_command_on_node $datafile" );  }
}
```

```
##External
#fvt_remote_shell_on_node_async
#
# Remote Shell On Node Asynchronous:
#   does exactly what fvt_remote_shell_on_node does, except that it does not
#   wait for the remote command to finish before returning.
#
# See &fvt_remote_shell_on_node for details.
#

sub fvt_remote_shell_on_node_async # ( $hostname_nodenum, $env,
$command, @script )
{
  local( $retVal, $tohost, $datafile ) = &fvt_remote_shell_on_node_prep
( @_ );
  if( $retVal == -1 )
    {  return -1;  }

  # Run the remote control program asynchronously
  if( &get_hostname eq $tohost )
    {  &quiet_system( "fvt_spawn_bg_process fvt_command_on_node
$datafile" );  }
  else
    {  &quiet_system( "fvt_spawn_bg_process fvt_remote_shell_path $tohost
fvt_command_on_node $datafile" );  }
}


##Internal
#fvt_remote_shell_on_node_prep
#
# Remote Shell On Node Prep
#   handles the processing of function parameters for
fvt_remote_shell_on_node*.
#   It checks for valid data, and creates a control file that
#   fvt_remote_shell_on_node* will use to pass its arguments to the remote
#   node.
#
# See &fvt_remote_shell_on_node for details.
#
sub fvt_remote_shell_on_node_prep # ( $hostname_nodenum, $env,
$command, @script )
{
  local( $host, $env, $command, @script ) = @_;
  local( $namepre, $nameext, $datafile ) = ( "$ENV{'HOME'}/.command.in",
0, "" );
  local( $tohost, $tonode, $internalEnv );

#  NOTES:
#  @script is used only by fvt_remote_db2_on_node to pass a db2 "script".
#  The contents of @script is always written to the control file, but
#  the software on the other side of the link only looks at it if
#  ($command eq "db2").

  # Get the host name and node number from the $host variable
  # Please note that for obvious reasons, a host name cannot contain only
  # digits...
  if( $host =~ /^\d+$/ )
  {
    $tohost = &fvt_node_to_name( $host );
    $tonode = $host;
  }
  else
  {
    $tohost = $host;
    $tonode = &fvt_name_to_node( $host );
  }
  unless( $tohost )
    {  return -1;  }

  # Grab the internal environment:
  $internalEnv = &fvt_append_env_var_list( "" );

  # Get a unique filename for passing the data...
  while (1)
  {
    $datafile = "$namepre.$nameext";
    if (-f $datafile )
      {  $nameext++;  }
    else
      {  last;  }
  }

  # Trim of new lines, if any...
  $tonode      =~ s/\n//g;
  $command     =~ s/\n//g;
  $internalEnv =~ s/\n//g;
  $env         =~ s/\n//g;
  foreach (@script )
    {  s/\n//g;  }

  # Write the data to the file...
  if( open( FILE, ">$datafile" ))
  {
    print FILE "$tonode\n";
    print FILE "$command\n";
    print FILE "$internalEnv $env\n";
    foreach (@script)
      {  print FILE "$_\n";  }
    close( FILE );
  }
  else
  {
    print "&fvt_remote_shell_on_node error: Could not open temporary file:
$datafile\n";
    return -1;
  }

  return ( 0, $tohost, $datafile );
}


##External
#fvt_reset_db_cfg
#
# macro to issue "db2 reset db cfg for ... " in a manner appropriate for
# both serial and MPP environments.  In serial, the command is issued as-is,
# in MPP it is issued using the &fvt_for_each_node() macro.
#
# usage:
#       $dbname = 'globaldb';
#       &fvt_reset_db_cfg( $dbname );
#
sub fvt_reset_db_cfg {

  local ( $dbname ) = shift;
  local ( $command );

  if ( &is_MPP() ) {
    ($nodes, @nodelist) = &fvt_db2nodes_manipulate("enum");
    foreach $node (@nodelist) {
      &fvt_remote_db2_on_node ($node, "TESTDIR = $TESTDIR",
                               "reset db cfg for $dbname",
                               "terminate");
    }
  }
  else {
    &db2("reset db cfg for $dbname", "terminate");
  }
  return;
}


##External
#fvt_run_on_all_nodes
#
# NOTE:  This macro must be used in conjunction with the fvt_createFileLock
()
#       function located in the fvt_comm function library.
#
# This macro runs an executable on all nodes in the db2nodes.cfg file
# simultaneously/asynchronously.  The output from each separate execution
# should match the expected result file.  If there exists a problem with
# the parameters passed to this macro, a return code of -1 will be generated
# and the macro will terminate; otherwise, a return code of 0 will be generated.
#
# Before exiting, this macro waits for each of it's child processes
# (testcases it started on remote nodes) to end and to create a file-lock
# using the fvt_createFileLock() function from the fvt_comm function library.
# To safeguard against the possibility of this macro waiting indefinitly for
# a child to complete, there is a maximum number of times that this macro will
# pole for the existence of the file-lock.  This number is 20 by default, but
# can be changed by the user (third parameter).  The duration of time between
# file-lock poles is 15 seconds.
```

```
#
# Syntax:  fvt_run_on_all_nodes( testcase_exe_name,
#                                rtest_sort_T_or_F
#                                [,max_num_poles/retries] );
#
# Example:  fvt_run_on_all_nodes( "testcase1", $TRUE, 10 );
#          - the value for $TRUE is defined within macro.pl
#          - the macro will wait a maximum time of 150 seconds
#            (10 retries * 15 seconds between retries)
#

sub fvt_run_on_all_nodes
{
   local( $testcase, $rtest_sort, $max_wait ) = @_;

   # Verify the receipt of arguments.
   if ( !defined( $testcase ) )
   {
      print "Error!  No executable name given!\n";
      return -1;
   }
   if ( !defined( $rtest_sort ) )
   {
      print "Error!  No sorting option given! [true/false]\n";
      return -1;
   }
   # If a value for the maximum number of times to query for the existence of
   # a program exit-file is not given, use the default value.
   if ( !defined( $max_wait ) )
   {
      $max_wait = 20;
   }
   print "Maximum number of retries is set to $max_wait.\n";

   # Set up the MPP environment variables.
   &get_cfg_nodes();

   local( $sleep_time ) = 15;
   local( $testdir ) = $ENV{TESTDIR};
   local( $num_nodes ) = $ENV{MPP_NODES};
   local( $node_number );
   local( $index );
   local( $pgm );
   local( $compare_opt );
   local( $remote_env_vars );
   local( $node_num_env_var );
   local( $rtest_cmd );

   print "\nRunning testcase $testcase on all nodes.\n";

   # Initial cleanup.
   &rm("$testdir/run/$testcase"."_*.*rn");
   &rm("$testdir/exp/$testcase"."_*.*xp");
   &rm("$testdir/exe/$testcase"."_*");

   for ($index=0; $index < $num_nodes; $index++)
   {
      # Get the node number.
      $node_num_env_var = sprintf( "MPP_NODE%4.4d", $index );
      $node_number = $ENV{$node_num_env_var};
      # Generate a program name.
      $pgm = "$testcase"."_"."$node_number";
      print "$pgm\n";
      # Create appropriatly named exe and exp files.
      &cp( "$testdir/exe/$testcase", "$testdir/exe/$pgm");
      &cp( "$testdir/exp/$testcase.pxp", "$testdir/exp/$pgm.pxp");
      &cp( "$testdir/exp/$testcase.rxp", "$testdir/exp/$pgm.rxp");
      # Determine which rtest comparison option to use.
      if ( $rtest_sort == $TRUE)
      {
         $compare_opt = "-S";
      }
      else
      {
         $compare_opt = "-R";
      }
      # Generate rtest command and invoke it on the currently selected node.
      $remote_env_vars = "TESTDIR=$testdir RUN_OPTS=$pgm";
      $rtest_cmd = "rtest $compare_opt $pgm";
      &fvt_remote_shell_on_node_async($node_number, $remote_env_vars,
$rtest_cmd);
   }

   # Wait for all child processes to terminate.
   #
   # Each testcase will create a file as it exits. The code below tests
   # for this file from each spawned child process to know when all
   # the child processes have completed.
   local( $done );
   local( $wait_count );

   print "Waiting for all concurrent testcases to finish\n";

   for ($index=0; $index < $num_nodes; $index++)
   {
      # Get the node number.
      $node_num_env_var = sprintf( "MPP_NODE%4.4d", $index );
      $node_number = $ENV{$node_num_env_var};
      # Generate a program name.
      $pgm = "$testcase"."_"."$node_number";
      $done = $FALSE;
      $wait_count = 0;
      # Check for the exit-files that should have been generated by the tc's.
      while ( $done == $FALSE )
      {
         # It the exit-file exists, the testcase has finished.
         if ( &fvt_checkFileLock($pgm) )
         {
            $done = $TRUE;
            if ( $wait_count == 0 )
            {
               print "$pgm is done.\n";
            }
            else
            {
               print "\n";
            }
         }
         else
         {
            if ( $wait_count == 0 )
            {
               print "Waiting for $pgm to complete";
            }
            $wait_count += 1;
            if ( $wait_count >= $max_wait )
            {
               print "\nERROR! Testcase is running too long. Not waiting any
longer\n";
               $done = $TRUE;
            }
            else
            {
               sleep $sleep_time;
               print ".";
            }
         }
      }
   }

   # Erase lock files.
   for ($index=0; $index < $num_nodes; $index++)
   {
      # Get the node number.
      $node_num_env_var = sprintf( "MPP_NODE%4.4d", $index );
      $node_number = $ENV{$node_num_env_var};
      # Generate a program name.
      $pgm = "$testcase"."_"."$node_number";
      # Remove the exit-file.
      &fvt_removeFileLock($pgm);
   }

   # Erase copies of the executables for this testcase.
   &rm("$testdir/exe/$testcase"."_*");
   #  Can't remove rxp files because even though the testcases have completed,
   #  the diff's are running right now.
   #  &rm("$testdir/exp/$testcase"."_*.*xp");
   return 0;
}


##External
#fvt_setup_autold_environment
#
```

```
# Currently only implemented for mpp.  Setups up the environment
# needed for the autoloader to work.
#
#  &fvt_setup_autold_environment
#

sub fvt_setup_autold_environment
{
   local ($password);


   $password = &get_password;

   &get_cfg_nodes;

   if ($AIX)
   {
      open (NETRC, "> $ENV{HOME}/.netrc");
      for ($i = 0; $i < $ENV{MPP_NODES}; $i ++ )
      {
         $envvar = sprintf( "MPP_NODE%4.4d_NAME", $i );
         $mach = $ENV{$envvar};
         print NETRC "machine $mach login $ENV{USER} password
$password\n";
      }
      close (NETRC);
      chmod 0600, "$ENV{HOME}/.netrc";
   }
}


##External
#fvt_update_db_cfg
#
# macro to issue "db2 update db cfg for ... " in a manner appropriate for
# both serial and MPP environments.  In serial, the command is issued as-is,
# in MPP it is issued using the &fvt_for_each_node() macro.
#
# usage:
#       $dbname = 'globaldb';
#       $parms = 'maxappls 40 avg_appls 20';
#       &fvt_update_db_cfg( $dbname, $parms );
#
sub fvt_update_db_cfg {

  local ( $dbname ) = shift;
  local ( $parms ) = shift;

  if ( &is_MPP() ) {
    ($nodes, @nodelist) = &fvt_db2nodes_manipulate("enum");
    foreach $node (@nodelist) {
      &fvt_remote_db2_on_node ($node, "TESTDIR=$TESTDIR",
                               "update db cfg for $dbname using $parms",
                               "terminate");
    }

  }
  else {
    &db2("update db cfg for $dbname using $parms", "terminate");
  }
  return;
}


##External
#get_cfg_nodes
#
# macro to get the nodenumber from db2nodes.cfg and set it within the
# environment for the driver and all the testcase using rtest in the
# driver could get it for usage.
#
# usage:
#       &get_cfg_nodes( );
#
# The nodenumber will be set in the environment as:
# MPP_NODE0000, MPP_NODE0001, ..., MPP_NODE(n-1) where n is the
number of nodes
# in the file, also, the number of nodes will be set in the environment
# variable MPP_NODES.  The macro will also retrun the number of nodes.
#
# In a similar fashion, the nodename will be set in the environment as:
# MPP_NODE0000_NAME, MPP_NODE0001_NAME, ...
#
# And, not to be outdone, the port number will be set in the environment as:
# MPP_NODE0000_PORT, MPP_NODE0001_PORT, ...
#

sub get_cfg_nodes
{
  local( $cfgname ) = &get_db2nodes_cfg( );
  local( $numnodes, $j, $value, $nodenum, $junk, $nodename, $nodeport );


  if (!-f($cfgname) )
  {
    die "can not open cfg file $cfgname\n";
  }

  open( NODES, $cfgname );
  @input = <NODES>;
  close(NODES);

  $numnodes = scalar(@input);
  $ENV{MPP_NODES} = $numnodes;

  for ($j = 0; $j < $numnodes; $j ++)
  {
    ( $nodenum, $nodename, $nodeport, $junk ) = split(/\s+/, $input[$j]);

    $temp = sprintf( "MPP_NODE%4.4d", $j );
#    &debug_printf( "set $temp to $nodenum" );
    $ENV{$temp} = $nodenum;

    $temp = sprintf( "MPP_NODE%4.4d_NAME", $j );
#    &debug_printf( "set $temp to $nodename" );
    $ENV{$temp} = $nodename;

    $temp = sprintf( "MPP_NODE%4.4d_PORT", $j );
#    &debug_printf( "set $temp to $nodeport" );
    $ENV{$temp} = $nodeport;
  }

  return( $numnodes );

}


##Internal
#get_db2nodes_cfg
#
# macro to get the complete path name of the db2nodes.cfg file
#

sub get_db2nodes_cfg
{
  local( $db2path ) = &get_db2path( );


  return( "$db2path/db2nodes.cfg" );
}


##External
#getnodetype
#
# macro to obtain the current node type
#
# This macro returns one of the following values:
#
#   STANDALONE
#   SERVER
#   REQUESTOR
#   STAND_REQ
#   MPP
#   UNKNOWN
#   ERROR: sqlfxsys returned XX,  sqlcode=NNNN
#
# example:
#    $nt = &getnodetype();
#
# One can set the environnement variable NODETYPE to ovverride.
# i.e.: if NODETYPE is set, it is returned by this function instead of
#       actually querying the nodetype.
```

```
sub getnodetype
{
   unless ($ENV{'NODETYPE'})
   {
      chop($ENV{'NODETYPE'} = `nodetype`);
   }

   if ($ENV{'NODETYPE'} =~ /ERROR/)
   {
     printf( "Error in determing nodetype : $ENV{'NODETYPE'}\n" );
     if (!$fvt_allow_nodetype_error)
     {
       exit 0;
     }
   }

   return $ENV{'NODETYPE'};
}


##External
#getsrvnodetype
#
# If we are running standalone, same as &getnodetype.
# If client-server, return the nodetype of the server.
#
# One can set the environnement variable SRVNODETYPE to ovverride.
# i.e.: if SRVNODETYPE is set, it is returned by this function instead of
#       actually querying the nodetype.
sub getsrvnodetype {
   local($nodetype);

   unless ($ENV{'SRVNODETYPE'})
   {
      chop( $nodetype = `nodetype` );
      if($nodetype eq "REQUESTOR")
      {
         $nodetype = &remote_shell($ENV{'CS_SERVER'},"nodetype");
      }
      $ENV{'SRVNODETYPE'} = $nodetype;
   }
   return $ENV{'SRVNODETYPE'};
}


##External
#is_MPP
#
#  Returns 1 if this is running on an MPP build
#

sub is_MPP
{
  local( $parallel );


  if (&getnodetype() eq "MPP")
  {
     $parallel = 1;
  }
  else
  {
     $parallel = 0;
  }

  return $parallel;

}


##External
#is_MPP_multinode
#
#  Returns 1 if this is running on an MPP build AND there is more than 1 node
defined
#

sub is_MPP_multinode
{
  local( $multi );


  if (&is_MPP && &get_cfg_nodes( ) > 1)
  {
     $multi = 1;
  }
  else
  {
     $multi = 0;
  }

  return $multi;

}


##External
#is_MPP_singlenode
#
#  Returns 1 if this is running on an MPP build AND there is only 1 node
defined
#

sub is_MPP_singlenode
{
  local( $single );


  if (&is_MPP && &get_cfg_nodes( ) == 1)
  {
     $single = 1;
  }
  else
  {
     $single = 0;
  }

  return $single;

}


##External
#notnfsdir
#
# macro to obtain the directory for creating a database
#
# This macro returns one of two possible values, NULL ("") if
# the database setup has no placement restrictions or
# a path to create the database on if there ARE restrictions
#

sub notnfsdir
{
   local($dbdir);


   if (&getsrvnodetype() eq "MPP")
   {
      $dbdir = $ENV{'NOTNFSDIR'};
      if ($dbdir eq "")
      {
        $dbdir = "/notnfs/$ENV{USER}";
      }
   }
   else
   {
      $dbdir = "";
   }

   return $dbdir;
}


##External
#remote_rtest
# Remote RTEST
#
# &remote_rtest( &lt;host_name | host_number&gt;, &lt;rtest_parameters&gt;,
#              string of (&lt;ENV_variable_name&gt;=&lt;its value&gt; ...) );
#
# Eg: &remote_rtest( 'hawk', "-pcR sample.sqc sample_db");
#     &remote_rtest( 'hawk', "-pcR sample.sqc sample_db",
#                  "HOME=$ENV{'HOME'} DBNAME=$DBNAME");
```

```
#

sub remote_rtest
{
  local($host) = shift;
  local($rtest_parms) = shift;
  local(@ENV_VARS) = @_;

  return( &fvt_remote_shell_on_node( $host, ($ENV{TESTDIR} ne "" ?
"TESTDIR=$ENV{TESTDIR} " : "") . "@ENV_VARS", "rtest
$rtest_parms" ));
}


##External
#set_cfg_node_numbers
#
# macro to alter db2nodes.cfg with a new set of node numbers
#
# usage:
#      &set_cfg_node_numbers( @array_of_new_numbers );
#
# example:
#
#      @foo = (100,200,300,400,500,600,700,800);
#      &set_cfg_node_numbers( @foo );
#

sub set_cfg_node_numbers
{
   local( @newnodes ) = @_;
   local( $cfgname, $cfgtemp, $j, $nodenum, $nodename, $mln, $route,
$numnodes );


   $cfgname = &get_db2nodes_cfg( );
   $cfgtemp = "db2nodes.cfg.temp";

   &cp( $cfgname, $cfgtemp );
   if (!-f( $cfgtemp ) )
   {
      die "temporary db2nodes.cfg file $cfgtemp could not be created\n";
   }

   open( NODES, $cfgtemp );
   @oldnodes = <NODES>;
   close( NODES );

   open( NODES, ">$cfgname" );

   $numnodes = scalar( @oldnodes );
   if ($numnodes > scalar( @newnodes ) )
   {
      printf( "not enough nodes supplied to set_cfg_node numbers, expected
at\n" );
      printf( "least $numnodes nodes\n" );
      die;
   }
   for ($j = 0; $j < $numnodes; $j ++)
   {
      ( $nodenum, $nodename, $mln, $route ) = split( /\s+/, $oldnodes[$j] );

#     &debug_printf( "change node $nodenum, $nodename, $mln, $route to
$newnodes[$j]" );
      $nodenum = $newnodes[$j];
      print NODES "$nodenum $nodename $mln $route\n";
   }

   close( NODES );

}


##Heading
#UDF Macros


##External
#load_byte_reverse_intel_udfs
# This subroutine loads 3 UDFs into the Database whose name
# is passed as the argument to the sub.
# These 3 UDFs use the fvt_byte_reverse_intel function in the
# fvt_comm.c file to byte reverse a HEX string only on INTEL
# platforms.
# byte4_rev_intel   -- is to byte reverse short ints
# byte8_rev_intel   -- is to byte reverse long ints
# byte16_rev_intel  -- is to byte reverse doubles
#
# Using these UDFs one can operate on the HEX value returned
# on shorts, long ints and doubles to produce output which
# is platform independent, yet does not mask out any wrong
# value returned by the HEX function.
#

sub load_byte_reverse_intel_udfs
{
  local($database) = @_;

  &get_sysval;
  $dllext = $sysval{'dllext'};
  $function_path = &sl( $sysval{'sqllibdir'}."/function" );

  # On AIX, presto the sqllib/function directory if necessary
  if( $AIX && ( -l $function_path ) )
  {
    &verbose_system("presto -d $function_path");
  }

  if( !( -f $function_path."/byte_rev".$dllext ) )
  {
    &cp( $sysval{'toolsdir'}."/byte_rev".$dllext,
$function_path."/byte_rev".$dllext );
  }

  &db2("connect to $database",
       "create function newton.byte4_rev_intel (varchar(4)) returns varchar(4)
external name 'byte_rev!byte_reverse_intel' language c parameter style db2sql
not fenced not variant no sql no external action",
       "create function newton.byte8_rev_intel (varchar(8)) returns varchar(8)
external name 'byte_rev!byte_reverse_intel' language c parameter style db2sql
not fenced not variant no sql no external action",
       "create function newton.byte16_rev_intel (varchar(16)) returns varchar
(16) external name 'byte_rev!byte_reverse_intel' language c parameter style
db2sql not fenced not variant no sql no external action",
       "connect reset",
       "terminate"
       );

  return;
}


##External
#load_round2_udf
# This sub load the round2 UDF into the database whose
# name is provided as the argument.
# This UDF uses the fvt_round function in fvt_comm.c
# to solve precision difference between platforms.
# The first parameter is the column (of type FLOAT) that needs rounding
# and the second parameter is the number of decimal places needed.
# The UDF returns a CHAR(22) in a platform independent form.
#
# macro to load round2 rounding UDF, takes
# the database name as an argument.
#

sub load_round2_udf
{
  local($database) = @_;

  &get_sysval;
  $dllext = $sysval{'dllext'};
  $function_path = &sl( $sysval{'sqllibdir'}."/function" );

  # On AIX, presto the sqllib/function directory if necessary
  if( $AIX && ( -l $function_path ) )
  {
    &verbose_system("presto -d $function_path");
  }

  if( !( -f $function_path."/round2".$dllext ) )
  {
    &cp( $sysval{'toolsdir'}."/round2".$dllext, $function_path."/round2".$dllext
);
```

```
  }

  &db2("connect to $database",
       "create function newton.round2 (float, int) returns char(22) external name
'round2!round2' language c parameter style db2sql not fenced not variant no
sql no external action",
       "connect reset",
       "terminate"
       );

  return;
}


##External
#load_round4_udf
# This sub load the round4 UDF into the database whose
# name is provided as the argument.
# This UDF uses the fvt_round function in fvt_comm.c
# to solve precision difference between platforms.
# The first parameter is the column (of type FLOAT) that needs rounding
# and the second parameter is the number of decimal places needed.
# The UDF returns a CHAR(22) in a platform independent form.
#
# macro to load round4 rounding UDF, takes
# the database name as an argument.
#

sub load_round4_udf
{
  local($database) = @_;

  &get_sysval;
  $dllext = $sysval{'dllext'};
  $function_path = &sl( $sysval{'sqllibdir'}."/function" );

  # On AIX, presto the sqllib/function directory if necessary
  if( $AIX && ( -l $function_path ) )
  {
    &verbose_system("presto -d $function_path");
  }

  if( !( -f $function_path."/round4".$dllext ) )
  {
    &cp( $sysval{'toolsdir'}."/round4".$dllext,
$function_path."/round4".$dllext );
  }

  &db2("connect to $database",
       "create function newton.round4 (real, int) returns char(22) external name
'round4!round4' language c parameter style db2sql not fenced not variant no
sql no external action",
       "connect reset",
       "terminate"
       );

  return;
}


##Heading
#Misc DB2 Macros


##External
#db2
#
# This macro enables DB2 commands to be issued in a standard fashion
without
# having to know platform specific details related to system call syntax.
#
# syntax:  &db2( [-<db2option>,] command1 [,command2] [,command3] ... );
#
# The DB2 macro can be passed one or more options that will be passed on
# to DB2.  (ex. +v, +c, ...)  The default options that are sent by &db2 at
# command invokation are +p, +t, and -v.
#
# Multiple DB2 commands can be given as parameters to this macro.  The
macro
# creates a temporary script file out of these parameters, which is executed by
# DB2.
#
# Example: &db2("-z outfile","+v",
#              "connect to mydb",
#              "list tables",
#              "select * from mytbl",
#              "connect reset",
#              "terminate" );
#
# If convenient, this macro can easily be extended to support different options
# for different commands (by translating for example, -v into
# "update command options using v ON" and putting it into the tmp file
# between the commands). This could prove useful with options such as -a.

sub db2
{
   local($ldb2opt,$lcmd,@cmd,$f,$outf);
   local($output) = @_ ;
   local($namepre,$nameext);

   if ( $output eq "#%" )
   {
     $output = 'y';
     shift ( @_ );
   }
   else
   {
     $output = 'n';
   }

   $ldb2opt = ""; #Init to avoid 'Use of unitialized variable' msg when using
perl -w.
   for (@_){
      if(/^\s*[\-\+]/){
         $ldb2opt .= " $_";
      } else {
         push(@cmd,$_);
      }
   }

   if ($ldb2opt!~/p/) {$ldb2opt .= " +p";}  # no interactive prompt

   #Acquire a unique name for the temporary script file.
   $namepre = "cmdtmp";
   $nameext = 0;
   while (1)
   {
      $f = "$namepre.$nameext.$$";
      if (-f $f)
      {
         $nameext++;
      }
      else
      {
         last;
      }
   }

   $outf = "cmdtmp.out";
   $lcmd = "db2 +t -v $ldb2opt -f $f";

   unlink($f);
   open(F,">$f") || print "&db2 error:Could not open temporary file:$f\n";
   for (@cmd) {print F "$_\n";}
   close(F);
   if($WIN || $WIN95)
   {
      $lcmd .= " -z $outf";
      unlink($outf);
   }

   if ( $output eq 'y' )
   {
      $rtn = &quiet_system_return($lcmd);
   }
   else
   {
      &quiet_system($lcmd);
   }

   if ($debug_no_execute)
   {
     # debug code - type out the file name if we are in debug mode
     &debug_printf( "===>" );
```

```
      &cat( $f );
      &debug_printf( "<===" );
   }

   unlink($f);
   if($WIN || $WIN95)
   {
      &cat($outf);
      unlink($outf);
   }

   if ( $output eq 'y' )
   {
      return $rtn;
   }

}


##Internal
#dflt_config_location
#
#   Determine default configuration files location.
#

sub dflt_config_location
{
   local($myrel) = &get_db2_ver();
   local($config_locn);

   if ($AIX || $UNIX)
   {
      $config_locn = "$ENV{'HOME'}/sqllib";
   }
   elsif ($OS2 || $WINT || $WIN )
   {
      if ($myrel ge "3")
      {
         $config_locn = "$ENV{DB2PATH}/cfg";
      }
      else
      {
         $config_locn = "$ENV{DB2PATH}";
      }
   }

   return $config_locn;
}


##Internal
#dflt_regr_config
#
#   Identify the default regression DB2 configuration file.  This file is used to
reset the
#   db2systm file before each bucket is run.
#
#   Example:
#      $nodenum = "";
#      dflt_regr_config($nodenum);
#

sub dflt_regr_config
{
  local( $nodenum ) = @_;
  local( $config_file );

  if ($nodenum eq "")
  {
      $config_file =  "db2syssr";
  }
  else
  {
      $config_file = "db2sysmp";
  }

  return $config_file;
}


##External
#forceagent
#
#    forceagent( userid, dbname )
#
#  Will put the PID of the agent for dbname.
#
#  This script asssumes that there is only one agent at the time the command is
issued.
#  Otherwise it won't work.

sub forceagent
{
  local( $userid, $dbname ) = @_ ;

  if ( $REAL_AIX || $OS2 ) {
     system("ps -ef | grep db2agent | grep $userid | grep $dbname | grep -v grep
| cut -c10-15 > forceit");
  } elsif ( $SunOS || $HPUX || $SINIXN || $SCO_SV || $UNIX_SV ) {
     system("db2 list applications | grep $ARGV[1] | cut -c31-35 >forceit");
  }

}


##External
#fvt_are_context_apis_available
#
# will return 1 if the current platfrom supports the context apis in db2
#

sub fvt_are_context_apis_available
{
  local( $retval ) = 1;


  # context apis are currently supported on all platforms except Unix ports

  if ($AIX)
  {
    if (!$REAL_AIX)
    {
      $retval = 0;
    }
  }

  return $retval;

}


##External
#fvt_can_chmod_directory
#
# will return 1 if the current platfrom allows a directory to be 'chmod'ed
#

sub fvt_can_chmod_directory
{
  local( $retval ) = 1;


  if ( ($OS2) || ($WINT) )
  {
    $retval = 0;
  }

  return $retval;

}


##External
#fvt_can_db2admin
#
#   Currently unix cannot do DB2ADMIN CREATE, while os2 and nt can,
#   and this extra power on intel needs its own tests.
#

sub fvt_can_db2admin
{
   if ($OS2 || $WINT)
   {
      return 1;
```

```
  }
  else
  {
     return 0;
  }
}


##External
#fvt_can_user_be_group
#
# will return 1 if the current platfrom allows a user and group to
# exist with the same name, otherwise will return 0.
#

sub fvt_can_user_be_group
{
  local( $retval ) = 1;


  if ($OS2 || $WINT)
  {
    $retval = 0;
  }

  return $retval;

}


##External
#fvt_commit_on_connect_reset
#
# indicates if DB2 on this platform will automatically commit when an
application
# issues a connect reset without doing an explicit commit
#

sub fvt_commit_on_connect_reset
{
  local( $result ) = 1;


  if ($WINT)
  {
    $result = 0;
  }

  return $result;
}


##External
#fvt_create_admin_server
#
#   This macro will create a db2 admin server.  On Intel - an actual instance is
#   created, while on Unix - a pointer is set up to an existing instance that
#   will function as the admin server
#

sub fvt_create_admin_server
{
  local( $password ) = &get_password( );


  if (&fvt_can_db2admin() )
  {
    &quiet_system( "DB2ADMIN CREATE /user:newton /
password:$password" );
  }
  else
  {
    $ENV{DB2ADMINSERVER} = "DB2DAS00";
  }
}


##External
#fvt_create_user_instance
#
# create an instance for this specific user, no parameters are supplied
#

sub fvt_create_user_instance
{
  local( $user );

  if ($OS2 || $WINT)
  {
    # since pconvert does not change the "regress" to the current user on intel
    # platforms - this macro must use "regress" as the user to create the
    # instance for.

    $user = "regress";

    &verbose_system( "db2icrt $user" );
    $ENV{DB2INSTANCE} = $user ;
  }
}


##External
#fvt_db2cli
#
# invoke db2cli on the indicated file, output will go to stdout, the input file
# must be in the test directory
#
# Example:
#
#    &fvt_db2cli( "tescase001" );
#

sub fvt_db2cli
{
  local( $test ) = @_;
  local( $db2path ) = &get_db2path( );


  if ($AIX )
  {
    &quiet_system( "$db2path/samples/cli/db2cli test/$test" );
  }
  else
  {
    &quiet_system( "$db2path\\samples\\cli\\db2cli test\\$test" );
  }
}


##External
#fvt_drop_user_instance
#
# drops an instance for this specific user, no parameters are supplied
#

sub fvt_drop_user_instance
{
  local( $user );


  if ($OS2 || $WINT)
  {
    # since pconvert does not change the "regress" to the current user on intel
    # platforms - this macro must use "regress" as the user to create the
    # instance for.

    $user = "regress";
    &verbose_system( "db2idrop $user" );
  }
}


##External
#fvt_get_lvl_file_path
#
# Returns the path (directory) where the "*.lvl" files are located.
# No parameters.
#

sub fvt_get_lvl_file_path
{
  local ($lvlpath) = &get_db2path;

  $lvlpath .= "/cfg";
```

```
  return $lvlpath;
}


##External
#fvt_get_nodelock_path
#
# Returns the path to the nodelock file
#
# Example: &fvt_get_nodelock_path();

sub fvt_get_nodelock_path {
  local ($release, $nodelockpath);
  $release = &get_db2_ver();
  if ($release lt "3") {
    printf("Error... unsupported release: $release.\n");
    die;
  }
  else {
    &get_sysval();
    $nodelockpath = $sysval{'nodelockpath'};
  }
  return $nodelockpath;
}


##External
#fvt_get_physical_db_path
#
# Input:   database name
# Output:  path to the database directory
# Example: get_physical_db_path("test")
#          returns /notnfs/rzheng/NODE0000/SQL00001
sub fvt_get_physical_db_path
{
  local($name) = @_;
  local($dbName);
  local($localDir);
  local($dbDir);
  local($nodeNum);
  local($skip);

  $name =~ tr/a-z/A-Z/;

  undef $dbName;
  undef $localDir;

  # Search the system directory for the database
  open(IN, "db2 list database directory |");
  while (<IN>) {
    if (!$dbName) {                    # Search for database name
      ($dbName) = /^\s+Database name\s+=\s+(.*)/;
      if ($dbName ne $name) {
        undef $dbName;
      }
    } else {                           # Grab local directory
      ($localDir) = /^\s+Local database directory\s+=\s+(.*)/;
      last;
    }
  }
  close(IN);

  if (!$dbName) {
    print "Database $name does not exist in the system directory\n";
    return "";
  }

  undef $dbName;
  undef $dbDir;
  undef $nodeNum;
  $skip = 4;

  # Now search the local directories for the database instance
  open(IN, "db2 list database directory on $localDir |");
  while (<IN>) {
    if (!$dbName) {                    # Again, search for DB name
      ($dbName) = /^\s+Database name\s+=\s+(.*)/;
      if ($dbName ne $name) {
        undef $dbName;
      }
    } elsif (!$dbDir) {                # Grab DB directory
      ($dbDir) = /^\s+Database directory\s+=\s+(.*)/;
    } elsif ($skip > 0) {              # Skip 4 lines to get to node num
      $skip--;
    } else {                           # Grab node number
      ($nodeNum) = /^\s+Node number\s+=\s+(.*)/;
      last;
    }
  }
  close(IN);

  if (!$dbName) {
    print "Database $name does not exist in the local directory\n";
    return "";
  }

  if ($WINT && !$WIN95) {              # NT localDir includes instance
    return(&sl(sprintf("$localDir/NODE%4.4d/$dbDir", $nodeNum)));
  } else {
    return(&sl(sprintf("$localDir/$ENV{DB2INSTANCE}/NODE%
4.4d/$dbDir", $nodeNum)));
  }
}


##External
#fvt_get_trybuy_file_path
#
# Returns the location (directory and filename) of the "Try & Buy" file.
# No parameters.
#

sub fvt_get_trybuy_file_path
{
  local ($tbfilepath) = &get_db2path;


  if ( ($OS2) | ($WINT) ) {
    $tbfilepath .= "/db2syslt";
  }
  else {
    $tbfilepath .= "/.netls/.db2syslt";
  }
  return $tbfilepath;
}


##External
#fvt_has_empty_local_db_directory
#
# returns an indication of the current platforms ability to have an
# empty database directory
#
# Example:
#
#    if (&fvt_has_empty_local_db_directory( ) )
#    {
#      do testcases ...
#    }
#

sub fvt_has_empty_local_db_directory
{
  local( $retval ) = 0;


  if (!$AIX)
  {
    $retval = 1;
  }

  return $retval;
}


##External
#fvt_has_make_fifo
#
# returns an indication of the current platforms ability to create a fifo
# queue via the make fifo command
#

sub fvt_has_make_fifo
```

```
{
  local( $retval ) = 0;


  if ($AIX)
  {
    $retval = 1;
  }
}


##External
#fvt_kill_dbm_then_app
#
#    Syntax:   &fvt_kill_dbm_then_app( "filename" );
#
# kill the database manager first and then kill any application using that
# database - the killdbm command can not guarantee to do this on every
platform
#
# The patamater is an optional filename of the output file, if omitted, stdout
# will be used
#

sub fvt_kill_dbm_then_app
{
  local( $file ) = @_;


  if ($AIX)
  {
    &fvt_system_with_output( "/wsdb/tools/killdbm", $file );

    # Really need to clean up ipc resources on hp/sun.

    if (!$REAL_AIX)
    {
      &fvt_system( "ipclean -a" );
    }
  }
  else
  {
    &fvt_system_with_output( "killdbm", $file );
  }
}


##External
#fvt_kill_license_daemon
#
# Kills the DB2 License daemon process (does NOT do a clean shutdown of
it).
#
# Example: &fvt_kill_license_daemon();
#

sub fvt_kill_license_daemon
{

  local($PID, $result, $ps_command, @result, @newresult, $killparm);
  local($kill_command) = "kill ";

  if ($WIN95) {
    $ps_command = "ps x | grep db2licd | grep -v grep |";
  }
  elsif ( ($OS2) || ($WINT) ) {
    $ps_command = "pstat | grep -i db2licd | grep -v grep |";
  }

  # Since "db2licd" is owned by root, it has to be killed using a "dbtsudo"
  # script.
  if ($AIX) {
    $kill_command = "dbtsudo fvt_netls_kill_db2licd.pl";
  }

  elsif ($OS2) {
    open(IN, "$ps_command");
    while (<IN>) {
      $PID = $_;
      chop $PID;
      $PID =~ s/^\s*(\w{4}).*/$1/;
      $PID =~ s/^\S*\s(\d*).*/$1/;
      if (length($PID) == 4) {last};
    }
    close(IN);
    # On OS/2, "pstat.exe" returns hex PIDs, but "kill.exe" expects decimal
    $PID = hex($PID);
  }
  elsif ($WINT) {
  # I had a beast of a time getting this to work on NT...
  #  the WinNT "grep.exe" would not work using "OPEN (... $command)".
  # I finally got it to work using back-ticks.
  #  Hence, a whole different approach for WinNT... LJM
    $result = `$ps_command`;
    @result = split("\n", $result);
    @newresult = grep (/pid:/, @result);
    @newresult = grep (/db2sysc/i, @newresult);
    $PID = shift (@newresult);
    $PID =~ s/^pid: (\d\d).*/$1/;
    $PID .= "\n";
  }

  $PID =~ s/^pid: (\d\d).*/$1/;

  system("$kill_command $killparm $PID");
  return;
}


##External
#fvt_can_I_kill_dbm_then_app
#
#    Syntax:   &fvt_can_I_kill_dbm_then_app();
#
# Since macro fvt_kill_dbm_then_app only work on UNIX platform now,
# it returns 1 and return 0 if it is in Intel platform.
# If later on, the macro become available in Intel platform, this
# routine will become always return 1 and become obselete.
#

sub fvt_can_I_kill_dbm_then_app
{
  if ($AIX)
  {
    return (1);
  }
  else
  {
    return (0);
  }
}


##External
#fvt_link_dbmig_msg
#
#    Syntax:   &fvt_link_dbmig_msg
#
#    This macro will link in the appropriate database migration message files so
that
#    database migration can run as if it is working from an installed image
#

sub fvt_link_dbmig_msg
{
  if ($AIX)
  {
    &verbose_system( "dbtsudo link_dbmigmsg" );
  }
}


##Internal
#fvt_modifyDBImageInfo
#
#  This macro is used by fvt_archive_dbdir and fvt_unarchive_dbdir to
recreate
# database instances.  This macro manipulates the contents of those images so
# that they can be used by a user other than the creator of the image (ie. the
# instance directory is altered, and the SQLTAGNAM tablespace container
# paths are updated).

sub fvt_modifyDBImageInfo
{
```

```
  local($instanceDir, $instanceID) = @_ ;

  local($curMacro) = 'fvt_modifyDBImageInfo';
  local($curDir);                    # Current directory
  local($filename);                  # Current file
  local(@dbDirs) = ();               # DBs in the instance directories
  local(@tagDirs) = ();              # TAG directories
  local(%sqlTagFiles);               # Files with tags, and their offsets
  local($debug) = 0;                 # Set =1 if debugging, =2 for detail

  local($tagFileName) = 'SQLTAG.NAM';    # Name of tag files
  local($initDir) = &get_curdir();
  local($i);                         # Loop counter
  undef(%sqlTagFiles);


  # ---------------------------------------------------------------------
  # Read the current instance directory and get the path of all DB directories
  # ---------------------------------------------------------------------
  if (!opendir(INSTDIR, "$instanceDir")) {
    die "$curMacro:  Failed to open directory $instanceDir\n";
  }
  chdir($instanceDir);               # Necessary for directory detection

  @dbDirs = readdir(INSTDIR);        # Get all files in instance dir
  closedir(INSTDIR);
  @dbDirs = grep(-d, @dbDirs);       # ... which are directories
  @dbDirs = grep(/^SQL\d+/i, @dbDirs);   # ... which begin with SQL[0-9]

  chdir('..');

  # Append complete path on database directory
  for ($i=0; $i<=$#dbDirs; $i++) {
    $dbDirs[$i] = &sl("$instanceDir/$dbDirs[$i]");
  }

  if ($debug) {print "[$curMacro]  Found DB's:\n";}
  if ($debug) {print join("\n", @dbDirs), "\n";}

  # ---------------------------------------------------------------------
  # Read each database directory to find SQLT and tsp entries
  # ---------------------------------------------------------------------
  foreach $dir (@dbDirs) {
    local(@dirList);                 # Master directory list
    local(@tspFiles);                # V2 TSP files in the DB directory

    # Read contents of database directory
    if (!opendir(DBDIR, $dir)) {die "$curMacro:  Failed to open $dir\n";}
    chdir($dir);                     # Necessary to qualify paths
    @dirList = readdir(DBDIR);

    # Fetch TSP files and DB directories
    @tspFiles = grep(-f, @dirList);
    @dirList = grep(-d, @dirList);

    # SQLS[0-9]+ directories... there will be TAGs here if this is a V1
    # database which has been migrated to V2
    foreach $segDir (grep(/SQLS\d/i, @dirList)) {
      local(@segDirList);            # Segment directories

      if (!opendir(SEGDIR, $segDir)) {die "$curMacro:  Failed to open
$segDir\n";}
      chdir($segDir);

      # Find SQLT[0-9]+ directories within the segment directory
      @segDirList = readdir(SEGDIR);
      @segDirList = grep(-d, @segDirList);
      @segDirList = grep(/SQLT\d/i, @segDirList);

      # Now push the relative directory path onto the master directory list
      foreach (@segDirList) {push(@dirList, &sl("$segDir/$_"));}

      chdir('..');
      closedir(SEGDIR);
    }

    # SQLT[0-9]+ directories
    foreach (grep(/SQLT\d/i, @dirList)) {push(@tagDirs, &sl("$dir/$_"));}

    # V2 TSP directories
    foreach (grep(/tsp\d/i, @dirList)) {push(@tagDirs, &sl("$dir/$_"));}

    # V2 TSP files - tags are embedded at offset 0x200 within
    foreach (grep(/tsp\d/i, @tspFiles)) {
      $sqlTagFiles{&sl("$dir/$_")}=0x200;
    }

    chdir($initDir);                 # Return to instance
    closedir(DBDIR);
  }

  if ($debug) {
    print "[$curMacro]  Found TAG directories:\n";
    print join("\n", @tagDirs), "\n";
    print "[$curMacro]  Found TSP files:\n";
    print join("\n", keys(%sqlTagFiles)), "\n";
  }

  # ---------------------------------------------------------------------
  # Read each SQL tag directory to find $tagFileName files
  # ---------------------------------------------------------------------
  foreach $tagFile (@tagDirs) {
    $tagFile = &sl("$tagFile/$tagFileName");

    # Check file permissions and create list of valid files
    if (!-f $tagFile) {die "$curMacro:  $tagFile does not exist";}
    if (!-r $tagFile) {die "$curMacro:  $tagFile not readable";}
    if (!-w $tagFile) {die "$curMacro:  $tagFile not writeable";}
    $sqlTagFiles{$tagFile} = 0;      # Tags are at start of SQLTAG.NAM
  }
  if ($debug) {
    print "[$curMacro]  Found SQLTAG's:\n";
    print join("\n", keys(%sqlTagFiles)), "\n";
  }

  # ---------------------------------------------------------------------
  # Now modify the contents of each of the tag files
  # ---------------------------------------------------------------------
  foreach (keys(%sqlTagFiles)) {
    local($F);                       # Current file contents
    local($ptr);                     # Pointer into $F
    local($len);                     # Length
    local($dbPath);                  # Path of database directory
    local($conTagVer);               # Container tag version
    local($checkSum);                # Checksum
    local($offset);                  # Offset of tag from start of file

    # Container tag constants
    local($tagSize)=512;             # Size of tag area
    local($instLen)=9;               # Length of instance ID in TAG
    local($instPos)=0x2f;            # Position of instance ID in TAG
    local($verLen)=4;                # TAG version length
    local($verPos)=8;                # TAG version position
    local($pathLen)=256;             # TAG container path length
    local($pathPos)=0x42;            # TAG container path position
    local($chksumLen)=4;             # TAG checksum length
    local($chksumV210Pos)=0x38;      # TAG checksum position before
V2.11
    local($chksumNewPos)=$tagSize-$chksumLen;
    local($chksumPos)=$chksumV210Pos;   # Default checksum position to
use


    # Feedback progress to user
    print "\t\t$_\n";

    # Get the files contents... binmode on necessary for non-UNIX
    undef $/;
    open(IN, $_);
    binmode(IN);
    seek(IN, $sqlTagFiles{$_}, 0);
    sysread(IN, $F, $tagSize);
    close(IN);
    $/ = "\n";

    if ($debug == 2) {&fvt_hexDump($F);}

    # ---------------------------------------------------------------
    # Update the stored instance ID and maybe the container paths if the
    # image is that of a new release.
    # See CONTAINER_TAG in sqlbistorage.h for definition of sizes used
below
    # ---------------------------------------------------------------
    # Update instance ID
```

```
    substr($F, $instPos, $instLen) = pack("a$instLen", $instanceID);

    # Fetch the container tag version number
    $conTagVer = unpack("L", substr($F, $verPos, $verLen));
    if ($debug == 2) {printf("Container tag version = %8.8lX\n",
$conTagVer);}

    # New (v2.1.1+) DB's have imbedded database paths
    if ($conTagVer > 0x212) {
      ($dbPath = $_) =~ s/(.*SQL\d\d\d\d\d.).*/$1/i;
      substr($F, $pathPos, $pathLen) = pack("a$pathLen", $dbPath);
      $chksumPos = $chksumNewPos;
    }

    # Compute new checksum as done in sqlbenvi.C::sqlbCheckSum()
    $checkSum = 0;
    for ($ptr=0; $ptr<$chksumPos; $ptr+=4) {# Read tag file 32-bits at a time
      $checkSum ^= unpack("L", substr($F, $ptr, 4));
    }
    substr($F, $chksumPos, $chksumLen) = pack("L", $checkSum);

    # Write the new tag file to disk
    open(OUT, "+<$_");
    binmode(OUT);
    seek(OUT, $sqlTagFiles{$_}, 0);
    syswrite(OUT, $F, $tagSize);
    close(OUT);

    if ($debug == 2) {
      print "Checksum: $checkSum\n";
      &fvt_hexDump(substr($F, $offset, $tagSize));
    }
  }

  chdir( $initDir );
}


##External
#fvt_named_pipes_as_protocol
#
#   Returns if the platform can use named pipes from one machine to
#   another.  (Currently only supported on nt)
#

sub fvt_named_pipes_as_protocol
{
   return ($WINT);
}


##External
#fvt_set_var_in_profile
#
# sets passed in environment variable in db2profile if it exists,
# otherwise in the environment
#

sub fvt_set_var_in_profile
{
  local($var,$setting)=@_;
  local( $profile );


  $profile = &get_db2path();
  $profile .= "/db2profile";

  if (-f $profile )
  {
    $addline1 = "${var}=${setting}";
    $addline2 = "export $var";
    open (TMPFILE, ">tempfile");
    print TMPFILE "$addline1 \n";
    print TMPFILE "$addline2 \n";
    close (TMPFILE);
    &cat( tempfile, $profile );
    &rm(tempfile);
  }
  else
  {
    $ENV{$var} = $setting;
  }
}


##External
#fvt_unarchive_dbdir
#
#  'fvt_unarchive_dbdir' is used to create a database from a single archive
# file, created by 'fvt_archive_dbdir'.
#
#  This macro will retrieve the databases stored in an archive file.  After the
# macro has returned, the caller will need to catalog the relelvant databases
# before they can be used.  Or, before the archive is recreated, the caller
# can create dummy databases which the archive contents can then be copied
# over.
#
#  Usage:
#     fvt_unarchive_dbdir [archiveName] [databasePath]
#     -- NOTES:  Do not include the archive's file extension.
#              Slashes are handled automatically.
#
#  Example:
#     &fvt_unarchive_dbdir("test/archive",
"/notnfs/krodger/krodger/NODE0001")

sub fvt_unarchive_dbdir {
  local($archivePath, $dbPath) = @_;

  $archivePath = &sl4(&sl($archivePath));
  $dbPath = &sl4(&sl($dbPath));

  # This macro has been replaced with the command line tool of the same
name
  system("fvt_unarchive_dbdir $archivePath $dbPath");
}


##External
#fvt_unlink_dbmig_msg
#
#    Syntax:   &fvt_unlink_dbmig_msg
#
#    This macro will undo the changes made in the fvt_link_dbmig_msg macro
#

sub fvt_unlink_dbmig_msg
{
  if ($AIX)
  {
    &verbose_system( "dbtsudo unlink_dbmigmsg" );
  }
}


##External
#fvt_v1_gateway_parmoldposition
#
# returns 1 if version 1 db2 on this platform ddcs gateway had
# parameters in old positions
#

sub fvt_v1_gateway_parmoldposition
{
  local( $retval ) = 0;


  if ($OS2)
  {
    $retval = 1;
  }
}


##External
#fvt_v1_existed
#
# returns 1 if version 1 db2 existed on this platform and needs to have
# migration tested
#

sub fvt_v1_existed
{
  local( $retval ) = 0;
```

```
  if ($OS2 || $REAL_AIX)
  {
    $retval = 1;
  }
}


##External
#fvt_v210_existed
#
# returns 1 if version 2.1.0 db2 existed on this platform and needs to have
# migration tested
#

sub fvt_v210_existed
{
  local( $retval ) = 0;


  if ($OS2 || $REAL_AIX || $WINT)
  {
    $retval = 1;
  }
}


##External
#fvt_v1_has_limited_protocols
#
# returns 1 if version 1 db2 on this platform only had very limited
# protocol support
#

sub fvt_v1_has_limited_protocols
{
  local( $retval ) = 0;


  if ($OS2)
  {
    $retval = 1;
  }
}


##External
#fvt_v1cae_not_merged
#
# returns 1 if version 1 cae directories were not merged in with the
# rest of the db2 directories
#

sub fvt_v1cae_not_merged
{
  local( $retval ) = 0;


  if ($OS2)
  {
    $retval = 1;
  }
}


##External
#get_db2path
#
# Returns the path to the root of the DB2 install directory
# (on UNIX, returns the path to sqllib;
#  on others, returns current DB2PATH environment variable setting).
#
# No arguments.
#

sub get_db2path
{
  if ($AIX)
  {
    return ("$ENV{HOME}/sqllib");
  }
  else
  {
    return ("$ENV{DB2PATH}");
  }
}


##External
#get_dbdir
#
# Returns the path where databases would be created by default.
# This function calls get_dbdir_no_db2instance() and then adds
# the db2 instance to the path if its applicable.
#
# No arguments.
#

sub get_dbdir
{
  local($dbdir);


  $dbdir = &get_dbdir_no_db2instance();

  if ($AIX)
  {
    $dbdir .= "/$ENV{DB2INSTANCE}";
  }
  else
  {
    if (&get_db2_ver ge "3")
    {
      $dbdir .= "\\$ENV{DB2INSTANCE}\\NODE0000";
    }
    elsif (&get_release eq "db2_v2")
    {
      if (!$OS2)
      {
        $dbdir .= "\\$ENV{DB2INSTANCE}";
      }
    }
  }

  if (!-d $dbdir)
  {
    &fvt_mkdir_r( $dbdir ) || die "could not create directory $dbdir\n";
  }

  return $dbdir;
}


#get_dbdir_no_db2instance
#
# Returns the path where databases would be created by default,
# but does not include the db2 instance in the path.
# No arguments.
#

sub get_dbdir_no_db2instance
{
  local($dbdir);


  if ($AIX)
  {
    if (&is_MPP( ) )
    {
      $dbdir = &notnfsdir( );
    }
    else
    {
      $dbdir = "$ENV{HOME}";
    }
  }
  else
  {
    if (&get_db2_ver ge "3" || &get_release eq "db2_v2")
    {
      $dbdir = "$ENV{SLAVEDRIVE}";
    }
  }
```

```
   return $dbdir;
}

##External
#get_dbdir_data
#
#     syntax: get_dbdir_data( version )
#
#    example: &get_dbdir_data( "db2_v1" );
#
# Gets information regarding the System and Local database directories,
#  modifies them so that they appear AS THEY WOULD HAVE APPEARED
in the
#  DB2 version passed to this macro, and assigns them to variables,
#  for example:
#
#    - variable '$system_dbdir_path' will contain the path to the
#       system database directory (e.g., 'c:\\sqllib\\' for Version 1 or 2
#       or 'c:\\sqllib\\[instancename]\\' on Version 3 on Intel platforms,
#       or '/sqllib/' for Unix platforms)
#
#    - variable '$local_dbdir_path' will contain the path to the
#       local database directory (e.g., 'c:\\' for Intel platforms
#       or '/sqllib/' for Unix platforms)
#
# Other variables also set are:
#
#    - '$local_dbdir_physical_path'
#    - '$local_dbdir_MPP_physical_path'
#    - '$default_db_path'
#

sub get_dbdir_data
{
  local( $version ) = @_;
  local( $db2instprof, $slavedrive );


  if ($AIX)
  {
    $default_db_path = $ENV{'HOME'};
    $system_dbdir_path = $ENV{'HOME'} . "/sqllib" ;
    $local_dbdir_path = $ENV{'HOME'};
    $local_dbdir_physical_path = $local_dbdir_path . "/" .
      $ENV{'DB2INSTANCE'};
    if ( $version ge "db2_v3" )
    {
      $local_dbdir_physical_path = $local_dbdir_physical_path .
"/NODE0000";
      $local_dbdir_MPP_physical_path = "/notnfs/$ENV{USER}/$ENV
{DB2INSTANCE}/NODE0000";
    }
  }
  else
  {
    $slavedrive = $ENV{SLAVEDRIVE};
    $slavedrive =~ tr/a-z/A-Z/;
    $db2instprof = $ENV{'DB2INSTPROF'};
    if ($db2instprof eq "") { $db2instprof = $ENV{'DB2PATH'}; }

    $local_dbdir_path = $slavedrive;
    $default_db_path = $slavedrive;

    if ( $version ge "db2_v2" )
    {
      $system_dbdir_path = $db2instprof . "/" . $ENV{'DB2INSTANCE'};
    }
    else
    {
      $system_dbdir_path = $db2instprof;
    }

    if ( $version lt "db2_v3" )
    {
      $local_dbdir_physical_path = $local_dbdir_path;
    }
    else
    {
      $local_dbdir_physical_path = $local_dbdir_path . "/" .
        $ENV{'DB2INSTANCE'} . "/NODE0000";
    }
  }

  return 0 ;
}


##Internal
#getcfg_location
#
#  Location of configuaration file db2systm.
#

sub getcfg_location
{
  local( $config_locn );

  if ($AIX || $UNIX)
  {
    $config_locn = "$ENV{'HOME'}/sqllib";
  }
  elsif ($OS2 || $WINT || $WIN )
  {
    $config_locn = $ENV{DB2INSTPROF};

    if ($config_locn eq "")
    {
      $config_locn = $ENV{DB2PATH};
    }

    $config_locn = $config_locn . "/" . $ENV{DB2INSTANCE};
  }

  return $config_locn;
}


##External
#killsys
#
#   killsys( )
#
#  Will bring down a database manager for the current user by
#  killing the system controller.  There are no parameters.
#

sub killsys
{
  local( $userid ) = $ENV{USER};
  local( $datafile ) ;
  local( $nameext ) = 0;
  local( $namepre ) = &get_hostname( );


  # Get a unique filename for passing the data...

  $namepre = "killsys.$namepre" ;
  while (1)
  {
    $datafile = "$namepre.$nameext";
    if (-f $datafile )
      {  $nameext++;  }
    else
      {  last;  }
  }
  if ($OS2) {
      $datafile .= ".cmd";
    }

  if ( $REAL_AIX ) {
    open(outfile, ">$datafile");
    print outfile "kill -9 ";
    close(outfile);
    &quiet_system("ps -ef | grep db2sysc | grep $userid | grep -v grep | cut
-c10-15 >> $datafile");
    &quiet_system("chmod u+x $datafile");
    &quiet_system("$datafile");
    &rm("$datafile");
  } elsif ( $AIX ) {
    open(outfile, ">$datafile");
    print outfile "kill -9 ";
    close(outfile);
    &quiet_system("ps -ef | grep db2sysc | grep $userid | grep -v grep | cut
```

```
-c10-15 | sort | head -1 >> $datafile");
     &quiet_system("chmod u+x $datafile");
     &quiet_system("$datafile");
     &rm("$datafile");
   } elsif ( $OS2 ) {
     open(outfile, ">$datafile");
     print outfile "kill ";
     close(outfile);
     &quiet_system("ps -ef | grep -i db2sysc | grep -v grep | cut -c3-7 >>
$datafile");
     &quiet_system("$datafile");
     &rm("$datafile");
   }

   return 0 ;
}


##External
#remove_backup
#
# remove a backup, parameter is the name of the database backup
# to remove.
#
#    example &remove_backup( "globaldb" );
#

sub remove_backup
{
   local( $dbname ) = @_;
   local( @files, $dbdir );

   $dbname =~ tr/a-z/A-Z/;

   opendir (CURDIR, '.');
   @files = readdir (CURDIR);
   closedir (CURDIR);
   foreach $dbdir ( @files )
   {
     if ( $dbdir =~ /^$dbname/ || ( $WINT && $dbdir =~ /^$dbname/i ) )
     {
       if ($AIX)
       {
         &rmdir( $dbdir );
       }
       elsif (-d $dbdir )
       {
         &rmdir( $dbdir );
       }
     }
   }
}


##Internal
#save_config
#
#   Save the local database configuration file as <config_file>.reg.
#   If the saved file already exists, then it will not
#   be overwritten or deleted.  This saved <config_file> will be used to reset
the
#   <config_file> before each new bucket is run.  This is required for SQLLIBs
that
#   have been created by a REAL install vs a development environment.
#
#   Example:
#      $nodenum = "";
#      save_config($node_num);
#

sub save_config
{
    local( $nodenum ) = @_;
    local( $config_locn, $dflt_config, $dflt_config_locn );

    $dflt_config = &dflt_regr_config($nodenum);
    $dflt_config_locn = &dflt_config_location();
    $config_locn = &getcfg_location();

    if ( !(-e "$dflt_config_locn/$dflt_config") && !(-e
"$config_locn/db2systm.reg") )
    {
      &cp("$config_locn/db2systm", "$config_locn/db2systm.reg");
      print "cp $config_locn/db2systm $config_locn/db2systm.reg\n";
    }
}


##External
#switch_config
#
#   Change the local database configuration.
#
#   Example:
#
#      &switch_config( "db2syssv" );
#

sub switch_config
{
    local($new_config) = @_ ;
    local ($diff) = 1;
    local ($dflt_config_locn, $config_locn);


    if ($debug_no_execute)
    {
      &debug_printf( "switch_config: setting config to $new_config" );
      return;
    }

    $dflt_config_locn = &dflt_config_location();
    $config_locn = &getcfg_location();

    if ( -e "$dflt_config_locn/$new_config" )
    {
      &cp("$dflt_config_locn/$new_config", "$config_locn/db2systm");
      $diff = &fvt_file_compare("$dflt_config_locn/$new_config",
"$config_locn/db2systm");
    }
    elsif ( -e "$config_locn/db2systm.reg" )
    {
      &cp("$config_locn/db2systm.reg", "$config_locn/db2systm");
      $diff = &fvt_file_compare("$config_locn/db2systm.reg",
"$config_locn/db2systm");
      printf( "could not find config file $new_config, used db2systm.reg
instead\n" );
    }
    elsif ( -e "$ENV{DB2PATH}/$new_config" )
    {
      &cp("$ENV{DB2PATH}/$new_config", "$ENV{DB2PATH}/db2systm");
      $diff = &fvt_file_compare("$ENV{DB2PATH}/$new_config", "$ENV
{DB2PATH}/db2systm");
      printf( "could not find config file $new_config in $$dflt_config_locn.\n");
      printf( "switched cfg files in $ENV{DB2PATH} directory instead.\n");
      printf( "This should only happen if you are doing down-level client/server
testing.\n" );
    }
    else
    {
      printf( "could not find config file $new_config to switch to\n" );
      exit (-1 );
    }

    if ( ! ($WIN95 || $WIN) )
    {
       # The default config files (db2sysrq, db2syssv, db2syssr, db2sysmp and
       # db2systm) are generated (by db2ftool) with a SYSADM_GROUP value
of "".
       # If "uselvl" is run, it explicitly updates "SYSADM_GROUP" to
"BUILD".
       # The tests expect "BUILD", so we have to set this back.
#       system("db2 +o update database manager configuration using
sysadm_group build");
       &db2( "+o update database manager configuration using sysadm_group
build", "terminate");
    }

    return $diff;
}


##Internal
#unsave_config
```

```
#
#   Remove the saved configuration file <config_file>.regr.
#   (see save_config for further info).
#

sub unsave_config
{
   local( $config_locn );

   $config_locn = &getcfg_location();

   if ( -e "$config_locn/db2systm.reg" )
   {
     &rm("$config_locn/db2systm.reg");
   }
}


##Heading
#Misc Macros


##Internal
#afsdir
#

sub afsdir {
   local($src, $type) = @_ ;

   &get_sysval( );

   if ($type eq "") { $type = substr( $sysval{'suffix'}, 1 ); }

   if ($type ne "aix")
   {
      $src =~ s/db2/db2$type/;
   }
   return($src);
}


##Internal
#build_bucket_name
#
# builds a bucket name from the information in owner.scr
#

sub build_bucket_name
{
  local( $bucket, $type, $class ) = @_;
  local( $temp );


  # Process $class to generate unique bucket entry
  # order is as follows:
  #  - dbcs
  #  - client/server
  #  - nameparm
  #  - MPP
  #  - SMP


  #
  # IMPORTANT
  #
  # If you change this here, make sure to take a look in the
fvt_command_2_bucket
  # macro to see if changes are necessary there.
  #
  #


  if ( $type =~ /D-/ )
  {
    ( $temp ) = ( $type =~ /D\-(\S\S)/ );
    $bucket .= "_$temp";
  }

  if ( $type =~ /CS-/ )
  {
    ( $temp ) = ( $type =~ /CS\-(\S\S)/ );
    $temp =~ tr/[A-Z]/[a-z]/;
    if ($temp eq "ai") { $temp = "ax"; }
    if ($temp eq "wi") { $temp = "nt"; }
    $bucket .= "_$temp";

    if ( $class =~ /mvs.+/ )
    {
      ( $temp ) = ( $class =~ /mvs(.)/ );
      $bucket .= $temp;
    }

    if ( $class =~ /mpp/ )
    {
      $bucket .= "p";
    }

    # now - do some extra stuff for downlevel client & server buckets

    if ( $class =~ /dlc.+v\d/ )
    {
      ( $temp ) = ( $class =~ /dlc.+v(\d)/ );
      $bucket .= "_dc$temp";
    }
    if ( $class =~ /dls.+v\d/ )
    {
      ( $temp ) = ( $class =~ /dls.+v(\d)/ );
      $bucket .= "_ds$temp";
    }

  }

  foreach( split(/\s+/, $class) )
  {
   $bucket .= "_$1" if (/^[p]\((.*)\)/);
  }

  if ( $type =~ /^P-/ && !( $class =~ /n\(/ ) )
  {
     # default for MPP buckets is one node if not specified

     $bucket .= "_n1";
  }

  foreach( split(/\s+/, $class) )
  {
   if (/^[n]\((.*)\)/)
   {
     $bucket .= "_n$1";
   }
  }

  foreach( split(/\s+/, $class) )
  {
   $bucket .= "_s$1" if (/^[s]\((.*)\)/);
  }

  return $bucket;
}


##Internal
#check_release
#

sub check_release
{
  local( $release ) = @_;
  local( $temprel ) = &get_release( );


  if ( $release ne $temprel )
  {
    printf( "The release $release is not valid for your current environment\n" );
    printf( "Your current environment release is $temprel\n" );
    exit 0;
  }

  return( 0 );
}


##Internal
#debug_printf
```

```
#
# This is actually a debug_print because of the way is was originally
# implemented. That is, all parameters are stuck into one string and
# printed instead of the first one being used as a format specifier
# for the rest.
sub debug_printf
{
   local( $outline ) = @_;
   local( $timestamp );


   $timestamp = &systime_stamp();
   if (!$debug_no_printf)
   {
      # If we used a printf here, it would try to interpret the string
      # as a format specifier and if it found something like '%s',
      # it would look for more parameters which were not provided and
      # therefore default to "" of 0.
      print( "$timestamp $outline\n" );
   }
}


##External
#find_and_cut
#
# &find_and_cut(<filename>, <string>, <lines>)
#
# find_and_cut takes 3 parameters, a filename, a string
# and a number of lines.  The subroutine matches the string
# to a line in the specified file and then cuts the number of
# lines specified.  It was designed to parse .rrn files in
# the ports.
#

sub find_and_cut
{
  local($i, $found);
  ($filename, $string, $num_to_cut) = @_;

  $i=0;
  $found="false";
  &cp("$filename TEMPFILE");
  open(INFILE, TEMPFILE);
  open(OUTFILE,"> $filename");
  while (<INFILE>)
     {
     if ($_ =~ /$string/)
        {
        $i=$num_to_cut;
        $found="true";
        }
     if ($i > 0)
        {
        $i = $i - 1;
        }
     else
        {
        print OUTFILE "$_";
        }
     }


  close INFILE;
  close OUTFILE;
  system("rm TEMPFILE");

}


##External
#fvt_begin_testcase
#
# print out standard "START OF TESTCASE .." message
#
#   example:  &fvt_begin_testcase( "a1065001" );
#

sub fvt_begin_testcase
{
  local( $testcaseName ) = @_ ;

  print("START OF TESTCASE: $testcaseName\n") ;
}


##External
#fvt_begin_unit
#
# print out standard "START OF TESTUNIT .." message
#
#   example:  &fvt_begin_unit( $testunit );
#

sub fvt_begin_unit
{
  local( $unitName ) = @_ ;

  print( "START OF TESTUNIT: $unitName\n") ;
}


##External
#fvt_db2trc_security_tested
#
#   Returns 'y' if db2trc on that platform is security tested.
#   Returns 'n' if db2trc can be turned on / off by any user
#   on that platform.
#
#   Example:
#
#      $db2trc_tested =  &fvt_db2trc_security_tested;
#

sub fvt_db2trc_security_tested
{
  if ( $AIX )
  {
    return 'y';
  }
  else
  {
   return 'n';
  }
}


##External
#fvt_end_testcase
#
# print out standard "TESTCASE xxx SUCCEEDED" message
#
#   example:  &fvt_end_testcase( "a1065001", "y" );
#

sub fvt_end_testcase
{
  local( $testcaseName, $successFlag ) = @_ ;

  if ( ! defined $successFlag )
  {
      print("TESTCASE: $testcaseName ENDED\n");
  }
  elsif ( $successFlag eq "y" )
  {
      print("TESTCASE: $testcaseName SUCCEEDED\n") ;
  }
  else
  {
      print("TESTCASE: $testcaseName FAILED\n") ;
  }

}


##External
#fvt_end_unit
#
# print out standard "TESTUNIT nn SUCEEDED" message
#
#   example:  &fvt_end_unit( $testunit, "y" );
#

sub fvt_end_unit
```

```
{
 local( $unitName, $successFlag ) = @_ ;

 if ( ! defined $successFlag )
 {
     print( "END OF TESTUNIT: $unitName\n") ;
 }
 elsif ( $successFlag eq "y" )
 {
    print( "TESTUNIT: $unitName SUCCEEDED\n");
    print( "END OF TESTUNIT: $unitName\n") ;
 }
 else
 {
    print( "TESTUNIT: $unitName FAILED\n");
    print( "END OF TESTUNIT: $unitName\n") ;
 }

}


##External
#fvt_fortran_250char_varfunc_supported
#
# This macro tests whether or not the Fortran compiler support a variable or
function name with 250 in length
#
sub fvt_fortran_250char_varfunc_supported
{
  local ($supported) = 1;
  if ($HPUX)
  {
    $supported = 0;
  }
  return ($supported);
}


# This macro tests whether or not the Fortran compiler support the PROCESS
directives
sub fvt_fortran_process_directives_supported
{
  local ($supported) = 0;
  if ($REAL_AIX)
  {
    $supported = 1;
  }
  return ($supported);
}

# This macro tests whether or not the Fortran compiler support variable names
th at begin with the underscore _
sub fvt_fortran_underscorevar_supported
{
  local ($supported) = 1;
  if ($HPUX)
  {
    $supported = 0;
  }
  return ($supported);
}

# This macro tests whether or not the Fortran compiler support variable names
that begin with $
sub fvt_fortran_dollarvar_supported
{
  local ($supported) = 1;
  if ($HPUX || $SunOS)
  {
    $supported = 0;
  }
  return ($supported);
}


##External
#fvt_fortran_double_quoted_strings_supported
#
# This macro will return 1 if the current platform fortran compiler supports
# strings quoted in double quotes
#
sub fvt_fortran_double_quoted_strings_supported
{
  local ($supported) = 1;

  if ($OS2)
  {
    $supported = 0;
  }
  return ($supported);
}


##External
#fvt_fortran_extended_keyword_byte_supported
#
# This macro will return 1 if the current platform fortran compiler supports
# the "byte" keyword
#

sub fvt_fortran_extended_keyword_byte_supported
{
  local ($supported) = 1;

  if ($OS2)
  {
    $supported = 0;
  }
  return ($supported);
}


##External
#fvt_fortran_long_var_names_supported
#
# This macro will return 1 if the current platform fortran compiler supports
# variable names longer than 30 characters.
#

sub fvt_fortran_long_var_names_supported
{
  local ($supported) = 1;

  if ($OS2 || $SunOS)
  {
    $supported = 0;
  }
  return ($supported);
}


##External
#fvt_has_rexx
#
#  Returns "1" if the current platform supports rexx, else returns "0".
#
#  Takes no arguments.
#
sub fvt_has_rexx {
  return ( ($OS2) || ($REAL_AIX) || ($WINT) );
}


##Internal
#fvt_hexDump
sub fvt_hexDump
{
  local($p) = @_;
  local(@array,$data,$len,$lstring,$offset);
  $lstring = $p;
  $data = $lstring;
  while (($len=length($lstring)) > 16)
  {
    $data    = substr($lstring,0,16);
    $lstring = substr($lstring,16);
    @array = unpack('N4', $data);
    $data =~ tr/\0-\37\177-\377/./;
    printf("%8.8lX  %8.8lX %8.8lX %8.8lX %8.8lX  %s\n",
          $offset, @array, $data);
    $offset += 16;
    $data = $lstring;
  }
```

```
  # Now finish up one byte at a time
  if ($len)
  {
    @array = unpack('C*',$data);
    $data =~ y/\0-\37\177-\377/./;
    for (@array) {$_ = sprintf('%2.2X',$_);}
    push(@array, '  ') while $len++ < 16;
    $data =~ s/[^ -~]/./g;
    printf("%8.8lX  %s%s%s%s %s%s%s%s %s%s%s%s %s%s%s%s  %s\n",
          $offset, @array, $data);
  }
}


##Internal
#fvt_interpret_range
#
# This function will take a numeric range and return the list of numbers that
# it specifies.
# NOTE: This macro is really simple.  It only handles positive integers!
# An error message will be printed to STDOUT if a range is not valid.
#
# Examples:
# &fvt_interpret_range( "3-6, 22, 47-46,2" ) -> ( 3, 4, 5, 6, 22, 46, 47, 2 )
# &fvt_interpret_range( "1,3,12-16" ) -> ( 1, 3, 12, 13, 14, 15, 16 )
# &fvt_interpret_range( "-39" ) -> ( ) + STDOUT ">>-39<< is not a valid
range.\n"
#
sub fvt_interpret_range # ( $range )
{
  local( $range ) = @_;
  local( @tokens, @subtokens, @range, $i );

  # Filter and check range for validity...
  $range =~ s/[^0-9,\-]//g;
  if( $range =~ /(^-)|(-$)|,,|,-|-,/ )
  {
    print ">>$range<< not a valid range.\n";
    return ();
  }

  @tokens = split( /,/, $range );

  foreach $token (@tokens)
  {
    @subtokens = split( /-/, $token );

    # Make sure the range is in the right order...
    if( (scalar( @subtokens ) > 1) && ($subtokens[0] > $subtokens[1]) )
    {
      $i = $subtokens[0];
      $subtokens[0] = $subtokens[1];
      $subtokens[1] = $i;
    }

    $range[scalar(@range)] = $subtokens[0];

    if( scalar( @subtokens ) > 1 )
    {
      for( $i=$subtokens[0]+1; $i<=$subtokens[1]; $i++ )
      {
        $range[scalar(@range)] = $i;
      }
    }
  }

  return @range;
}


##External
#fvt_is_debug_build
#
# Macro will return "1" if the current build is a debug build,
# or "0" if it is an optimized build, or "-1" if it cannot
# determine either way.
#
# Please note that you *MUST* be already CONNECTed to a database
# in order to use this macro.
#
# Arguments: none
#
# Syntax: &fvt_is_debug_build();
#
sub fvt_is_debug_build
{
  local ($db2_debug, $ctr);
  undef($db2_debug);
  $ctr = 0;
  &fvt_redirect_output("is_debug.out");
  system("db2 .opt");
  &fvt_restore_output;

  # if sqlcode = 0 then build is debug
  #  else build is optimized
  # Debug: DB20000I
  # Optimized: SQL0104N and DB20000I
  open(LOGFILE, "< is_debug.out");

  while (<LOGFILE>) {
    shift;
    $ctr++;
    if (/SQL0104N/) {
      $db2_debug = "N";
      }
    } # while (<LOGFILE>)...

  if ( ($db2_debug eq " ") && ($ctr ne 4) ) {
    print "Incorrect number of lines ($ctr) returned (should be 4).\n";
    $db2_debug = -1;
    }
  elsif ($db2_debug ne "N") {
      $db2_debug = 1;
      }
    else {
      $db2_debug = 0;
      }

  close(LOGFILE);
  &rm('is_debug.out');
  return $db2_debug;
}


##External
#fvt_linker_export_support
#
#returns 1 if the current platform's link editor support the ability
#to export symbols or define default entry points for symbol table
#

sub fvt_linker_export_support
{
   local ($doexport) = 1;
   if ($AIX && (!$REAL_AIX))
   {
     $doexport = 0;
   }
   return ($doexport);
}


##External
#fvt_mask_range
#
#    Syntax:   &fvt_mask_range("command", "key", "start", "stop");
#
#    Executes a program and masks out a specified portion of any
#    lines of output which contain a specified key character string.
#
#    For example,
#
#       &fvt_mask_range("db2licd -q", "Peak DB2 Users:", "20", "40");
#
#    would print all of the output from the "db2licd -q" command
#    *EXCEPT* that all text between (and including) columns 20 and 40
#    of any line containing "Peak DB2 Users:" would be replaced by "*"s.
#
#
sub fvt_mask_range {
  local($program, $key, $start, $stop, $range, $mask, $string_length,
*SAVEOUT, *SAVEERR);
  $program = shift;
  $key = shift;
```

```
  $start = shift;
  $stop = shift;
  $range = $stop - $start + 1;
  for (1 .. $range) {
    $mask .= "*";
  }

  if ($start < 1)
  {
    $start = 1;
  }

  #Redirect STDOUT and STDERR to a file
  &fvt_redirect_output("redirect.txt");

  #Execute the command
  system("$program");

  #Restore STDOUT and STDERR to normal
  &fvt_restore_output();

  #Look thru output line by line
  open(REDIRFILE, "<redirect.txt");
  while ($_ = <REDIRFILE>) {
    if (/\.*$key\.*/) {
      $string_length = length($_);
      substr($_, $start - 1, $range) = $mask;
      if ($string_length <= $stop) {
        $_ .= "\n";
      }
    }
    print;
  }
  close(REDIRFILE);

  #Then, delete the file.
  &rm("redirect.txt");
}


##External
#fvt_redirect_output
#
#    Syntax:   &fvt_redirect_output("filename");
#
#    Redirects STDOUT annd STDERR to a specified file.
#
#    For example,
#
#       &fvt_redirect_output("redirect.txt");
#
#    would redirect STDOUT and STDERR to a file called "redirect.out".
#
#    STDOUT and STDERR can be restored to normal using &restore.
#
sub fvt_redirect_output {
  local($file);
  $file = shift;
  open(SAVEOUT, ">&STDOUT");
  open(SAVEERR, ">&STDERR");

  open(STDOUT, ">$file") || die "Can't redirect STDOUT.";
  open(STDERR, ">&STDOUT") || die "Can't dup STDOUT.";

  select(STDOUT); $| = 1;   # Make unbuffered
  select(STDERR); $| = 1;   # Make unbuffered
}


##External
#fvt_redirect_output_null
#
#    Syntax:   &fvt_redirect_output_null( )
#
#    Redirects STDOUT annd STDERR to null
#
sub fvt_redirect_output_null
{
  &fvt_redirect_output( $null );
}


##External
#fvt_restore_output
#
#    Syntax:   &fvt_restore_output;
#
#    Restores STDOUT and STDERR to what they were before they were
#    redirected to a file using the &redirect macro.
#
#    For example,
#
#       &fvt_restore_output;
#
#    would return STDOUT and STDERR to their previous settings.
#
#
sub fvt_restore_output {
  close(STDOUT);
  close(STDERR);

  open(STDOUT, ">&SAVEOUT");
  open(STDERR, ">&SAVEERR");
}


##External
#fvt_system
#
# fvt_system: Performs the command, after doing any platform specific
#      forward to back slash conversions
#

sub fvt_system
{
  local( $cmd ) = @_;


  $cmd = &sl4( &sl( $cmd ) );
  &quiet_system( $cmd );
}


##External
#fvt_system_with_output
#
# fvt_system_with_output: Performs the command and place output in
#      the specified file.
#
# &fvt_system_with_output( "command", "output file" );
#
#   command: the command to be done by the operating system.
#   filename: Name of the file where the output is redirected.
#            Caution that the contents of this file, if it already existed
#            would be overwritten.
#

sub fvt_system_with_output
{
  local ($cmd0, $file ) = @_;


  if ($file ne "")
  {
    open(SAVEOUT, ">&STDOUT");
    open(SAVEERR, ">&STDERR");
    open(STDOUT, ">$file");
    open(STDERR, ">$file");
  }

  &quiet_system($cmd0);

  if ($file ne "")
  {
    close(STDOUT);
    close(STDERR);
    open(STDOUT, ">&SAVEOUT");
    open(STDERR, ">&SAVEERR");
  }
}


##External
#fvt_testOldSqlApis
```

```
#
# Returns FLASE on platforms where the SQL limits cannot
# be tested due to some limitation (like compiler limitation)
#
# Example: if ( &fvt_testOldSqlApis ) {
#            system("rtest -pcR c1084ca1.c"); }
#

sub fvt_testOldSqlApis
{
  if ($WIN95 || $WINT || $WIN)
  {
    return 0;
  }
  return 1;
}

##External
#fvt_testSqlLimitInPackage
#
# Returns FLASE on platforms where the SQL limits cannot
# be tested due to some limitation (like compiler limitation)
#
# Example: if ( &fvt_testSqlLimitInPackage ) {
#            system("rtest -pcR csbcp006.sqc globaldb"); }
#

sub fvt_testSqlLimitInPackage
{
  if ($WIN95 || $WINT || $WIN)
  {
    return 0;
  }
  return 1;
}


##External
#fvt_tests_abnormal_termination
#
#  Returns 1 if running on a platform on which abnormal process termination
#  is to be tested.  Else, returns 0.
#

sub fvt_tests_abnormal_termination
{
   return ($AIX==1);
}


##Internal
#get_blduname
#
# This function offers the correct behaviour (as opposed to get_uname).
# e.g.: the previous one could return different strings for different SINIX
machines.
#       it would also differ between AIX 3.2.5 and AIX 4.1
#       (it also returned HP-UX but HPUX is more consistent with the way we
use it) etc.
# The "uname" function is a system utility on UNIX systems so we dont have
control over
# the different outputs/formats it give whereas the "blduname" is a tool in
wsdb.
# What needs to be done is to grep for get_uname everywhere and update all
the matches
# to use get_blduname; then delete get_uname.
# Any volunteers welcome :)
# See Ben Martel and/or Hoang Duong about this.

sub get_blduname {

  if ($OS2)
  {
    $system_name = "OS2";
  }
  elsif ($WIN)
  {
    $system_name = "WIN";
  }
  elsif ($WINT)
  {
    # The environment variable "winbootdir" should be defined by Windows
    # on Windows 95 but not on Windows NT
    if($ENV{winbootdir} ne "")
    {
      $system_name = "WIN95";
    }
    elsif($ENV{Cpu} eq "PPC")
    {
      $system_name = "WINTPPC";
    }
    else
    {
      $system_name = "WINT";
    }
  }
  else
  {
    chop($system_name = `blduname`);
  }
  $system_name;
}


##Internal
#get_groupdir
#
# macro to parse the passed in test directory name and return the
# group (or type) to which the test belongs.
#
# The group (or type) is something like standalone, dbcs_stand.Ja_JP, etc.
#

sub get_groupdir
{
   local( $directory ) = @_;
   local( $groupdir );


   ( $groupdir ) = ( $directory =~ /\S*\/\S*\/(\S*)\/\S*/ );
   if ($groupdir eq "dbcs_stand")
   {
      $groupdir = "dbcs_stand.Ja_JP";
   }

   if (!$groupdir)
   {
     $groupdir = "unknown";
   }

   return $groupdir;
}


##External
#get_hostname
#
# Return the current tcpip host name
#
# There are two related macros:
#
# - get_hostname_db2 - return the hostname, but remove any characters db2
#   does not like (eg. '-')
#
# - get_hostname_unique - get the host name suffixed with the uid, used on
#   unix platforms to get a unique host for each id
#

sub get_hostname
{
   if ( $AIX )
   {
      if ( $REAL_AIX )
      {
         chop ($MachineName=`hostname -s`) ;
      }
      else
      {
         chop ($MachineName=`hostname`);
      }
   }
   elsif( $UNIX_SV)
   {
      chop ($MachineName=`/usr/ucb/hostname`) ;
```

```
  }
  else
  {
     chop ($MachineName=`hostname -s`) ;
  }
  $MachineName =~ tr/[A-Z]/[a-z]/;
  if ($OS2 || $SCO_SV)
  {
    # on OS/2, we get <host>.torolab.ibm.com, and we only want the <host>
    # portion

    $MachineName =~ s/\..*//;
  }

  return( $MachineName );
}


##External
#get_hostname_db2
#
# Return the current tcpip host name, but remove any characters db2 does
# not like (eg. '-')
#

sub get_hostname_db2
{
  local ( $MachineName );


  $MachineName = &get_hostname;
  $MachineName =~ s/\-//g;

  return( $MachineName );
}


##Internal
#get_hostname_unique
#
# Return a unique host name for this host and uid combination, it is the
# tcpip host name, suffixed with the uid number.
#

sub get_hostname_unique
{
  local ( $MachineName );


  $MachineName = &get_hostname;
  if ( $AIX )
  {
     # issue 'id' command & get UID
     chop ($id = `id`);
     $id =~ s/uid\=(\d+).*/$1/;
     # append padded UID to end of $MachineName
     $id = sprintf( "%04d", $id );
     $MachineName = $MachineName . $id;
  }

  return( $MachineName );
}


##Internal
#get_level
#

sub get_level
{
   local( $level, $levelfile );


   if ($OS2 || $WINT || $WIN)
   {
      $levelfile = "$ENV{SLAVEDRIVE}\\level";
      if (!-f $levelfile)
      {
         printf( "unable to determine level, file $levelfile not found\n" );
         die;
      }
      chop( $level = `type $levelfile` );
      $level =~ s/ //g;
   }
   else
   {
      chop( $level = `bldenv TOP` );
      $level =~ s/.*db2.*\_v\d+\//;
   }
   return( $level );
}


##Internal
#get_parms
#
# Subroutine to parse the command line.
# It puts in @args the command line arguments.
# It puts in %opts the command line options.
# An option is of the form (-|/)<opt>[(:|=)<value>]
# An option with no value is assigned 1.
# All option names are lowercased.

sub get_parms
{
   local($key,$value);

   undef %opts; undef @args; #wipe out anything there.
   while($_ = shift @ARGV)
   {
      if(s/^[\/\-]//)      #Is it an option?
      {
         if (substr( $_, 0, 1) eq "\/" || substr( $_, 0, 1) eq "\-")
         {
            push(@args,$_);
         }
         else
         {
            if (/\=/)
            {
               ($key, $value) =  split(/\=/,$_,2);
            }
            else
            {
               ($key, $value) =  split(/:/,$_,2);
            }

            $key =~ tr/[A-Z]/[a-z]/; #Not case senitive for opts.
            if ( $value eq "" ) { $value = 1; }
            $opts{$key} = $value;
         }
      }
      else              #Otherwise its an arg.
      {
         push(@args,$_);
      }
   }
}


##External
#get_password
#
#   This will return the current password for the regression id the bucket
#   is currently running on.  It will not return a password for a non-
#   regression id.
#
#   $password = &get_password( );
#

sub get_password
{
   local($password,$bldpath);

   if($AIX) {
      $bldpath = "$ENV{HOME}/sqllib/int/TOP";
   } elsif($WINT || $WIN || $OS2) {
      $bldpath = "$ENV{SRCTOP}";
   } else {
      die "get_password: Error unknown OS.\n";
   }
   open(TEST_PASS,"<${bldpath}/test/test_pwd") || die "get_password: Error
cant open ${bldpath}/test/test_pwd.\n";
  chop($password = <TEST_PASS>);
```

```
  close(TEST_PASS);
  return $password;
}


##Internal
#get_release
#

sub get_release
{
  local( $rel );


  if ($AIX)
  {
    # unix - use bldenv and get the release

    chop( $rel = `/wsdb/tools/bldenv RELEASE` );
    #parse out platform specific stuff:
    $rel =~ s/(db2).*(_v.)/$1$2/;
  }
  else
  {
    # intel - parse it from the scrtop env variable

    $rel = $ENV{SRCTOP};
    $rel =~ tr/[A-Z]/[a-z]/;
    $rel =~ s/.*(db2\_\S\S).*/$1/;
  }

  if ($rel eq "db2_v500")
  {
    $rel = "db2_v3";
  }

  return( $rel );
}


##Internal
#get_db2_ver
#

sub get_db2_ver
{
  local( $rel );


  $rel = &get_release( );
  ( $rel ) = ( $rel =~ /db2_v(\d)\d*/ );

  if (!$rel)
  {
    printf( "unable to determine numeric db2 release\n" );
  }

  return( $rel );
}


##Internal
#get_sysval
#
#Get system specific values.

sub get_sysval
{
   local($blduname);

   $blduname = &get_blduname;

   if($OS2)
   {
      %sysval = ('bldpath',      $ENV{SRCTOP},
                 'sqllibdir',    $ENV{DB2PATH},
                 'suffix',       '.os2',
                 'objext',       '.obj',
                 'libext',       '.lib',
                 'exeext',       '.exe',
                 'dllext',       '.dll',
                 'objdir',       'obj',
                 'libdir',       'lib',
                 'db2_v2',       'db2_v2',
                 'db2_v3',       'db2_v3',
                 'nodelockpath', "$ENV{'I4_INSTALL_DRIVE'}/ifor/ls/conf",
                 'optionparm',   '/',
                 'toolsdir',     "$ENV{SRCTOP}/test/fvt/standalone/tools",
                 );
   }
   if($WIN){
      %sysval = ('bldpath',      $ENV{SRCTOP},
                 'sqllibdir',    $ENV{DB2PATH},
                 'suffix',       '.win',
                 'objext',       '.obj',
                 'libext',       '.lib',
                 'exeext',       '.exe',
                 'dllext',       '.dll',
                 'objdir',       'obj',
                 'libdir',       'lib',
                 'db2_v2',       'db2_v2',
                 'db2_v3',       'db2_v3',
                 'optionparm',   '/',
                 'toolsdir',     "$ENV{SRCTOP}/test/fvt/sa/tools",
                 );
   }
   if($WINT){
      %sysval = ('bldpath',      $ENV{SRCTOP},
                 'sqllibdir',    $ENV{DB2PATH},
                 'suffix',       '.wint',
                 'objext',       '.obj',
                 'libext',       '.lib',
                 'exeext',       '.exe',
                 'dllext',       '.dll',
                 'objdir',       'wintobj',
                 'libdir',       'wintlib',
                 'db2_v2',       'db2_v2',
                 'db2_v3',       'db2_v3',
                 'nodelockpath', "$ENV{'I4_INSTALL_DRIVE'}/ifor/ls/conf",
                 'optionparm',   '/',
                 'toolsdir',     "$ENV{SRCTOP}/test/fvt/standalone/winttools",
                 );

      if($WINTPPC)
      {
        # WindowsNT PPC uses a different tools directory

        $sysval{'suffix'} = ".wintppc";
        $sysval{'objdir'} = "wintppcobj";
        $sysval{'libdir'} = "wintppclib";
        $sysval{'toolsdir'} = "$sysval{bldpath}/
test/fvt/standalone/wintppctools";
      }

      if($blduname eq "WIN95")
      {
        # Windows 95 uses a different tools directory

        $sysval{'suffix'} = ".win95";
        $sysval{'toolsdir'} = "$sysval{bldpath}/test/fvt/standalone/w95tools";
      }
   }
   if($AIX)
   {
      %sysval = ('bldpath',      "$ENV{HOME}/sqllib/int/TOP",
                 'sqllibdir',    "$ENV{HOME}/sqllib",
                 'suffix',       '',
                 'objext',       '.o',
                 'libext',       '.a',
                 'exeext',       '',
                 'dllext',       '',
                 'objdir',       'obj',
                 'libdir',       'lib',
                 'db2_v2',       'db2_v2',
                 'db2_v3',       'db2_v3',
                 'nodelockpath', '/usr/lib/netls/conf',
                 'optionparm',   '-',
                 'toolsdir',     "$ENV{HOME}/
sqllib/int/TOP/test/fvt/standalone/tools",
                 );
      if($blduname eq "SunOS")
      {
        $sysval{'db2_v2'} = 'db2sun_v2';
        $sysval{'db2_v3'} = 'db2sun_v3';
```

```
        $sysval{'suffix'} = '.sun';
        $sysval{'nodelockpath'} = '/var/netls';
      }
      elsif($blduname eq "HPUX")
      {
        $sysval{'db2_v2'} = 'db2hp_v2';
        $sysval{'db2_v3'} = 'db2hp_v3';
        $sysval{'suffix'} = '.hp';
        $sysval{'nodelockpath'} = '/usr/netls/nodelock';
      }
  }
  $sysval{'uname'} = &get_uname;
  $sysval{'blduname'} = $blduname;

  if($WIN){
      $sysval{sa_dir} = "sa";
      $sysval{cs_dir} = "sa";
  } else {
      $sysval{sa_dir} = "standalone";
      $sysval{cs_dir} = "standalone";
  }

  return %sysval;
}


##Internal
#get_uname
#

sub get_uname {

  if ($OS2)
  {
    $system_name = "OS2";
  }
  elsif ($WIN)
  {
    $system_name = "WIN";
  }
  elsif ($WINT)
  {
    # The environment variable "winbootdir" should be defined by Windows
    # on Windows 95 but not on Windows NT
    if($ENV{winbootdir} ne "")
    {
      $system_name = "WIN95";
    }
    elsif($ENV{Cpu} eq "PPC")
    {
      $system_name = "WINTPPC";
    }
    else
    {
      $system_name = "WINT";
    }
  }
  else
  {
    chop($system_name = `uname`);
  }
  $system_name;
}


##External
#has_sqlbind
#
#   This function will return true if the current platform supports the
#   sqlbind function
#
#   if (&has_sqlbind( ) )
#   {
#       ... perform sqlbind testcase(s)
#   }
#

sub has_sqlbind
{
  if ( $OS2 || $AIX )
  {
    return 1;
  }
  else
  {
    return 0;
  }
}


##External
#keep_version
#
#  args: path name, suffix
#  use:  appends a suffix to the end of all files, or files relating to a specified
#        testcase, in a given directory (e.g., run or err directories) so that they
#        will not be overwritten by subsequent re-runs of the same testcase.
#
#  example1: &keep_version( "run", "1" ) ;
#
#            This would append ".1" to the end of all files in the ".../run"
#            directory (e.g.: "f4testcs.rrn" would become "f4testcs.rrn.1").
#
#  example2: &keep_version( "run/f4testcs", "1" ) ;
#
#            This would append ".1" only to the end of the files for testcase
"f4testcs"
#            in the ".../run".

sub keep_version {
  local($_,$suffix) = @_ ;

  if (/^(.*)\/(.*)$/)
    {
      $dir=$1;
      $testcasename=$2;
    }
  else
    {
      $dir=$_;
      $testcasename="";
    }

  $prefix="";
  $working_dir = "$ENV{TESTDIR}/$dir";

  opendir(DIR,"$working_dir");
  local(@modfiles) = readdir(DIR);
  closedir(DIR);

  for $prefix (@modfiles)
  {
      next if $prefix eq '';
      next if $prefix eq '.';
      next if $prefix eq '..';
      next if ($prefix =~ /.*\..*\..*/ ); #ignore those with >1 extensions
      next if ($testcasename ne "") && ($prefix !~ /$testcasename/);
      &rn($working_dir,$prefix,$prefix.".".$suffix);
  }

}


##External
#mkdirrec
#
#    Given a pathname, make the subdirectories of the path it they do not exist.
#

sub mkdirrec
{
  local( $path ) = @_;
  local( $i, $temp_path, $delim );


  if ( ! -d $path )
  {
    @splitpath = split(/[\/,\\]/, $path);
    $temp_path = "/";
    $i = 1;
    if ($splitpath[0])
    {
      $i = 0;
      $temp_path = "";
```

```
    }

    if (!$AIX)
    {
      if ( $splitpath[0] =~ /:/ )
      {
        $temp_path = $splitpath[0] . "/";
        $i = 1;
      }
    }

    for (; $i <= $#splitpath; $i++)
    {
      $temp_path .= $splitpath[$i] . "/";

      if ( ! -d $temp_path )
      {
        mkdir( $temp_path, 0777 );
      }
    }
  }
}


##Internal
#mkcopydir
#
#    Identify the local path to store the err, run and res results of a bucket into.
#

sub mkcopydir
{
  local( $release, $level, $buckettype, $bucketdir, $copytodir ) = @_;
  local( $homedir, $copydir );

  if ($copytodir eq "")
  {
    $homedir = &get_db2path();
    $homedir =~ s/[\\\/]sqllib//;
    $homedir .= &sl("/RR");
  }
  else
  {
    $homedir = $copytodir;
  }

  $copydir = "$homedir/$release/$level/$buckettype/$bucketdir";

  return $copydir;
}


##Internal
#mkafsdir
#

sub mkafsdir
{
  local( $rel, $level, $type, $bucket ) = @_;
  local( $afsroot ) = &mkafsroot( $rel );

  return( "$afsroot/$level/$type/$bucket" );
}


##Internal
#mkafsroot
#

sub mkafsroot
{
  local( $rel ) = @_;

  return( "/afs/tor/groups/dbp/fvt_runs/$rel" );
}


##Internal
#mkrepdir
#

sub mkrepdir
{
  local( $rel, $local ) = @_;
  local( $pathname );


  if ($local)
  {
    $homedir = &get_db2path();
    $homedir =~ s/[\\\/]sqllib//;

    $pathname = "$homedir/dbmtest/runreport/$rel";
  }
  else
  {
    $pathname = "/u/dbmtest/runreport/$rel";
  }
  $pathname .= "/standalone";

  return $pathname;
}


##Internal
#mkrepfile
#

sub mkrepfile
{
  local( $rel, $level, $local ) = @_;
  local( $pathname );


  $pathname = &mkrepdir( $rel, $local);
  $pathname .= "/$level";

  return $pathname;
}


##Internal
#build_runreport_line
#

sub build_runreport_line
{
  local( $corner, $bucket, $machine, $pass, $fail, $time, $defect, $comment )
= @_;

  $temp  = sprintf( "$corner%-$runreport_width{bucket}.$runreport_width
{bucket}s", $bucket );
  $temp .= sprintf( "$corner%-$runreport_width{machine}.$runreport_width
{machine}s", $machine );
  $temp .= sprintf( "$corner%$runreport_width{pass}.$runreport_width{pass}
s", $pass );
  $temp .= sprintf( "$corner%$runreport_width{fail}.$runreport_width{fail}
s", $fail );
  $temp .= sprintf( "$corner%-$runreport_width{'time'}.$runreport_width
{'time'}s", $time );
  $temp .= sprintf( "$corner%-$runreport_width{defect}.$runreport_width
{defect}s", $defect );
  $temp .= sprintf( "$corner%-s", $comment );

  return $temp;
}


##Internal
#output_runreport_line
#

sub output_runreport_line
{
  local( $file, $corner, $bucket, $machine, $pass, $fail, $time, $defect,
$comment ) = @_;


  printf( $file &build_runreport_line( $corner, $bucket, $machine, $pass, $fail,
$time,
     $defect, $comment ) );
  printf( $file "\n" );

}
```

```
##Internal
#quiet_system
#

sub quiet_system
{
  local($command) = @_;


  if ($debug_yes_printf || $debug_no_execute)
  {
    print "$command\n";
  }

  if (!$debug_no_execute)
  {
    system $command;
  }

}


##Internal
#quiet_system_return
#

sub quiet_system_return
{
  local($command) = @_;
  local( $return );
  local( $namepre ) = 'cmdtext';
  local( $nameext ) = 0;
  local( $cmdfile );


  if ($debug_yes_printf || $debug_no_execute)
  {
    print "$command\n";
  }

  $return = "";
  if (!$debug_no_execute)
  {
    #
    # Temporary kludge For WinNT (backtick DB2 commands hang).... LJM
05/09/97
    #

    if ($WINT) {
      while(1) {
        $cmdfile = $namepre.$nameext;
        if (-f $cmdfile) {
          $nameext++;
        }
        else {
          last;
        }
      }
      &fvt_system_with_output("$command", $cmdfile);
      open(TMPFILE, $cmdfile);
      while(<TMPFILE>) {
         $return .= $_;
      }
      close (TMPFILE);
      &rm($cmdfile);
    }
    else
    {
      $return = `$command`;
    }
  }

  return $return;

}


##External
#remote_shell
#
# Execute a command remotely.
# Usage: &remote_shell(host, command [,user [,password] ]);
# If user and password are not specified, the default is to
# use the current user and the regression password.
#
sub remote_shell
{
  local( $host, $command, $rsh_user, $password ) = @_;
  local( $localuser ) = "-n";
  local( *RSH_SESSION, $rsh_output, $tmpfile, $cmd, $return );

  # provide useful defaults for user and password.
  if(!$password)
  {
    $password = &get_password();
  }
  if(!$rsh_user)
  {
    $rsh_user = $ENV{USER};
  }
  if(!$rsh_user)
  {
    die "USER environment variable not set up, remote_shell macro unable to
continue\n";
  }

  if ( $OS2 )
  {
    $localuser="-u $ENV{USER}";
  }
  if ( $WIN )
  {
    $localuser="-u regress";
  }
  if ( $WIN95 )
  {
     $localuser="-p $password";
  }
  if ( $AIX )
  {
     $localuser="";
  }

  $return = &quiet_system_return( "$rshcmd $host -l $rsh_user $localuser
$command" );

  return $return;
}


##Internal
#remote_shell_hawk
#
sub remote_shell_hawk
{
  local( $cmd ) = @_;
  local( $return );


  $return = &remote_shell_hawk_return( $cmd );
  print "$return";
}


##Internal
#remote_shell_hawk_return
#
sub remote_shell_hawk_return
{
  local( $cmd ) = @_;
  local( $return );


  &debug_printf( "remote_shell_hawk $cmd" );
  $return = &remote_shell( "hawk", "bin/run_cron $cmd", "regress" );
  return $return;
}


##External
#remote_shell_path
#
```

```
# Execute a command remotely, with the path set up on the
# host end.  See the remote_shell macro for a description of
# all parameter options.
#
# Usage: &remote_shell_path(host, command [,user [,password] ]);
#

sub remote_shell_path
{
  local( $host, $command, $rsh_user, $password ) = @_;
  local( $return );


  $return = &remote_shell_path_output( $host, $command, $rsh_user,
$password );
  print $return;
}


##External
#remote_shell_path_output
#
# Perform the same operation as remote_shell_path, but return
# the output.
#
# Usage: $output = &remote_shell_path_output(host, command [,user
[,password] ]);
#

sub remote_shell_path_output
{
  local( $host, $command, $rsh_user, $password ) = @_;
  local( $return );


  # if db2 instance is specifed - pass it over to remote side

  if ($ENV{DB2INSTANCE} ne "" )
  {
    $command = "-i$ENV{DB2INSTANCE} $command";
  }

  $command = ".regression_rsh_wrap $command";

  $return = &remote_shell( $host, $command, $rsh_user, $password );
  return $return;
}


##Internal
#systime_stamp
#
# Used by verbose_system to print a time stamp on a command

sub systime_stamp
{
   local($hour,$min,$sec,$junk);
   ($sec,$min,$hour,$junk,$junk,$junk,$junk,$junk,$junk) = localtime(time);
   return (sprintf ("%02d", $hour) ":". sprintf ("%02d", $min) ":". sprintf ("%
02d", $sec));
}


##External
#trusted
#
# Returns true if the current OS runs a trusted client.
# An optional parameter can be used to query the trustability
# of an OS in particular.
# e.g.: &trusted() and &trusted("OS2") are the same only when
#       you are running under OS2.

sub trusted
{
   local($os) = @_;
   if($os && $os ne "WIN95" && $os ne "WIN"){
      return 1;
   } elsif(!$WIN95 && !$WIN) {
      return 1;
   } else {
      return 0;
   }
}


##Internal
#uncompress_dir_name
#

sub uncompress_dir_name
{
  local( $directory ) = @_;


  if (substr( $directory, 0, 1) eq "\@" )
  {
    # we have a compressed directory - expand it out

    printf( "compressed directory $directory, " );
    if (substr( $directory, 1, 2) eq "cs" )
    {
      $directory = "test/fvt/client_server".substr( $directory, 3 );
    }
    elsif (substr( $directory, 1, 4) eq "dbcs" )
    {
      $directory = "test/fvt/dbcs_stand".substr( $directory, 5 );
    }
    else
    {
      $directory = "test/fvt/standalone".substr( $directory, 2 );
    }
    printf( "expanded to $directory\n" );
  }

  return( $directory );

}


##Internal
#verbose_system
#

sub verbose_system {
   local($command) = @_;
   local( $rc ) = 0;


   &debug_printf($command);
   if (!$debug_no_execute)
   {
     $rc = system( $command ) >> 0;
   }

   return $rc;
}


##Internal
#verbose_system_return
#

sub verbose_system_return {
   local($command) = @_;
   local( $return );


   &debug_printf($command);
   if (!$debug_no_execute)
   {
     $return = `$command`;
   }

   return $return;
}


##Heading
#Compile and Link

##External
#In Testcases
#
# All compiling and linking in testcases should be done using rtest.
```

```
#
# The 'c' option of rtest will compile an source file into an executable
# The 'l' option will compile a source file into a library suitable for
#   loading as a udf
# The 'o' option will compile a source file into an object than can be
#   linked with other objects.  Simply add this object into the C_OBJS_PRE
#   or .._POST environment variable prior to the link.
#
# For rtest or rbkt specific help, please refer to their entries off
# of the regression homepage.
#


##Internal
#ccompile
#
# Compile and Link

sub ccompile
{
  local( $file, $useropts, $outfile, $setuidroot, $reentrant ) = @_;
  local($osopts, $oslibs, $ldb2);


  if (!$outfile) { $outfile = $file; }

  $file = &sl4(&sl($file));
  $outfile = &sl4(&sl($outfile));
  $useropts = &sl4(&sl($useropts));

  if ($AIX)
  {
    $idir = &getincdir;
    $cc = "cc";
    $oslibs = "";
    if ($setuidroot)    # don't include the db2 library for setuidroot programs
    {
       $ldb2 = "";
    }
    else
    {
       $ldb2 = "-ldb2";
    }

    if( $REAL_AIX )
    {
       $osopts = "-qcpluscmt -DSQL_REALAIX ";
       if ($reentrant && $REAL_AIX4)
       {
          $cc = "cc_r";
          $osopts .= "-DSQLAIX4 ";
       }
    }
    if ($SunOS)
    {
       $osopts = "-xCC -DSQLSUN -DHANDLE_MISALIGNED_DATA
-Dpascal= -D_loadds= -Dfar=";
    }
    if ($HPUX)
    {
       if($straight_c_override ne "yes") {$cc = "CC";}
       $osopts = "-w -Aa +a1 -D_HPUX_SOURCE -DHPUX -Dpascal=
-D_loadds= -Dfar=";
       if ($ldb2 =~ "db2")    # Ken says this is necessary
       {
          $ldb2 .= " -lhppa ";
       }
    }
    if ($SINIXN)
    {
       $osopts = "+a1 -W0 +d -Kr4000 -DSQLSNI -Dpascal= -D_loadds=
-Dfar=";
    }
    if ($SCO_SV)
    {
       $osopts = "-DSQLSCO -DSQLOpenServer -Dpascal= -D_loadds=
-Dfar=";
       $oslibs = "-lsocket";
    }
    if ($UNIX_SV)
    {
       $osopts = "-DSQLSCO -DSQLUnixWare -Dpascal= -D_loadds= -Dfar=";
       $oslibs = "-lsocket -lm -lgen -lresolv";
    }
    &verbose_system( "$cc -o $outfile$ext -g -DSQLAIX -DSQLUNIX
-DSQLZDEBUG -DLINT_ARGS -I$idir/INST/include -L$idir/INST/lib
$osopts $useropts $file.c $ldb2 $oslibs" );
  }
  if ($OS2)
  {
    $os2libs = "db2api.lib";
    &verbose_system( "icc $useropts -Ti -Fo$outfile.obj -Fe$outfile$ext
$os2libs -Ss -DSQLOS2 -DSQLZDEBUG -DLINT_ARGS /B\"/MAP\" /
B\"/NOE\" /B\"/BAT\" /B\"/ST:64000\"  $file.c" );
  }
  if ($WINT)
  {
    # The doas tool requires a DB2 API call, thus we need db2api.lib
    $win32libs = "db2api.lib kernel32.lib advapi32.lib user32.lib";

    if($WIN95)
    {
       &verbose_system( "cl $useropts -DSQLWINT -DSQLWIN95 -Z7 -Fo
$outfile.obj -Fe$outfile$ext -Tc$file.c $win32libs /link /DEBUG /
PDB:NONE");
    }
    elsif ($WINTPPC)
    {
       &verbose_system( "cl $useropts -DSQLWINT -DSQLWINTPPC -Z7 -Fo
$outfile.obj -Fe$outfile$ext -Tc$file.c $win32libs /link /DEBUG /
PDB:NONE");
    }
    else
    {
       &verbose_system( "cl $useropts -DSQLWINT -Z7 -Fo$outfile.obj -Fe
$outfile$ext -Tc$file.c $win32libs /link /DEBUG /PDB:NONE");
    }
  }
  if ($WIN)
  {
    open(OUT, ">buildos.bat") || warn "Can't open buildos.bat";
    print OUT "set PATH=t:\\msvc\\bin;%PATH%\n";
    print OUT "set
INCLUDE=t:\\dos\\db2_v3\\include;t:\\dos\\tcpdos\\include\\rsa;.;t:\\msvc\\inc
lude;%INCLUDE%\n";
    print OUT "set LIB=t:\\dos\\tcpdos\\lib;t:\\msvc\\lib;%LIB%\n";
    print OUT "set CL=-Zi -O \/DSQLWIN \/DDB2WIN \/D_WINDOWS
\/DNEED_EXTERNS \/AL \/Zp4\n";
    print OUT "cl -c -Fo$outfile.obj -I$ENV{SRCTOP}\\test\\fvt\\inc -I. -Tc
$file.c > $file.out\n";
    if (!($useropts =~ /-c/))
    {
       print OUT "link $outfile.obj,$outfile.exe,nul,\/NOD \/NOE llibce
oldnames socketl \/stack:32000\;\n";
    }
    close(OUT);
    &make_rexx_file("buildos.cmd","buildos.bat");
    &verbose_system("buildos.cmd");
    &rm("buildos.bat");
    &rm("buildos.cmd");
  }
}


##Internal
#cdllcompile
#

sub cdllcompile
{
  local( $file, $outfile ) = @_;
  local($useropts, $osopts, $cmd);


  if (!$outfile) { $outfile = $file; }

  $file = &sl4(&sl($file));
  $outfile = &sl4(&sl($outfile));
  $useropts = &sl4(&sl($opt));

  if ($AIX)
  {
    $idir = &getincdir();
    if($REAL_AIX)
```

```
  {
    $cc = "xlC";
    $osopts = "-qcpluscmt -bE:$file.exp -bM:SRE -e$outfile";
  }
  if ($SunOS)
  {
    $cc = "cc";
    $osopts = "-xCC -G -DSQLSUN -DHANDLE_MISALIGNED_DATA";
  }
  if ($HPUX)
  {
    $cc = "CC";
    if($straight_c_override eq "yes") {$cc = "cc";}
    $osopts = "+Z -w -Aa +a1 -DHPUX -c ";
    $ext = ".o";
  }
  if ($SINIXN)
  {
    $cc = "CC";
    $osopts = "-G -DSQLSNI ";
  }
  if ($SCO_SV)
  {
    $cc = "cc";
    $osopts = "-G -DSQLSCO -DSQLOpenServer";
  }
  if ($UNIX_SV)
  {
    $cc = "cc";
    $osopts = "-G -DSQLSCO -DSQLUnixWare";
  }
  $cmd = "$cc $osopts -o $outfile$ext ../../../obj/fvt_comm.o -I../../../inc ";
  $cmd .= "-I../../../../../INST/include -g -lm -Dpascal= -D_loadds= -Dfar= ";
  $cmd .= "-DSQLAIX -DSQLUNIX -DSQLZDEBUG -DLINT_ARGS
-DSQLTOAIX -DSQEXTERNC=\'extern ";
  $cmd .= "\"c\"\' $useropts $file.c -L$idir/INST/lib -ldb2";
  &verbose_system("$cmd");
  if ($HPUX)  # need 2nd stage for HP
  {
    $cmd = "ld -b -E ../../../obj/fvt_comm.o ../../../lib/fvtlib.a ../../../
lib/dbcs_lib.a ";
    $cmd .= "-L../../../lib -L/usr/ccs/lib -L../../../../../../../lib -L$idir/INST/lib
-L/db2/hplib -L/usr/lib ";
    $cmd .= "-lm -lcurses -lnetls -lnck -lX11 -lXt -lXm -lcl -lsec -lhppa -ldb2
-o$outfile $outfile.o";
    &verbose_system("$cmd");
  }
 }
 if ($OS2)
 {
  &verbose_system( "useorcreatedef .\\ $file" );
  $cmd = "icc $useropts -Ge- -O- -Ti -Fo$outfile.obj -Fe$outfile$ext
db2api.lib -Ss -DSQLOS2 ";
  $cmd .= "-DSQLZDEBUG -DLINT_ARGS /B\"/MAP\" /B\"/NOE\" /
B\"/BATCH\" /B\"/ST:64000\" ";
  $cmd .= "-I..\\\\..\\\\..\\\\inc ..\\\\..\\\\..\\\\obj\\\\fvt_comm.obj $file.c $file.def"
;
  &verbose_system("$cmd");
 }
 if ($WINT)
 {
  &verbose_system( "useorcreatedef .\\ $file" );
  # The doas tool requires a DB2 API call, thus we need db2api.lib
  $win32libs = "db2api.lib kernel32.lib advapi32.lib user32.lib";

  if($WIN95)
  {
    $cmd = "cl $useropts -O -GX -J -Z7 -DWIN32_LEAN_AND_MEAN
-DDB2WINT -DSQLWINT -DSQLWIN95 ";
    $cmd .= "/DLL /Fe$outfile -I$extractpath\\test\\fvt\\inc -I$sysval
{sqllibdir}\\include ";
    $cmd .= "$file.c ..\\..\\..\\obj\\fvt_comm.obj
$extractpath\\test\\fvt\\lib\\fvtlib.lib ";
    $cmd .= "$sysval{sqllibdir}\\lib\\db2api.lib $sysval{sqllibdir}
\\lib\\db2cli.lib ";
    $cmd .= "..\\..\\..\\lib\\dbcs_lib.lib /LD /link /NODEFAULTLIB:libc /
DEBUG /PDB:NONE";
    &verbose_system("$cmd");
  }
  elsif($WINTPPC)
  {
    $cmd = "cl $useropts -O -GX -J -Z7 -DWIN32_LEAN_AND_MEAN
-DDB2WINT -DSQLWINT -DSQLWINTPPC ";
    $cmd .= "/DLL /Fe$outfile -I$extractpath\\test\\fvt\\inc -I$sysval
{sqllibdir}\\include $file.c";
    $cmd .= "..\\..\\..\\obj\\fvt_comm.obj $extractpath\\test\\fvt\\lib\\fvtlib.lib
$sysval{sqllibdir}\\lib\\db2api.lib ";
    $cmd .= "$sysval{sqllibdir}\\lib\\db2cli.lib ..\\..\\..\\lib\\dbcs_lib.lib /LD /
link /NODEFAULTLIB:libc /DEBUG /PDB:NONE";
    &verbose_system("$cmd");
  }
  else
  {
    $cmd = "cl $useropts -O -GX -J -Z7 -DWIN32_LEAN_AND_MEAN
-DDB2WINT -DSQLWINT /DLL /Fe$outfile ";
    $cmd .= "-I$extractpath\\test\\fvt\\inc -I$sysval{sqllibdir}\\include
$file.c ..\\..\\..\\obj\\fvt_comm.obj ";
    $cmd .= "$extractpath\\test\\fvt\\lib\\fvtlib.lib $sysval{sqllibdir}
\\lib\\db2api.lib $sysval{sqllibdir}\\lib\\db2cli.lib ";
    $cmd .= "..\\..\\..\\lib\\dbcs_lib.lib /LD /link /NODEFAULTLIB:libc /
DEBUG /PDB:NONE /DEF:$file.def";
    &verbose_system("$cmd");
  }
 }
 if ($WIN)
 {
  open(OUT, ">buildos.bat") || warn "Can't open buildos.bat";
  print OUT "set PATH=t:\\msvc\\bin;%PATH%\n";
  print OUT "set
INCLUDE=t:\\dos\\db2_v3\\include;t:\\dos\\tcpdos\\include\\rsa;.;t:\\msvc\\inc
lude;%INCLUDE%\n";
  print OUT "set LIB=t:\\dos\\tcpdos\\lib;t:\\msvc\\lib;%LIB%\n";
  print OUT "set CL=-Zi -O \/DSQLWIN \/DDB2WIN \/D_WINDOWS
\/DNEED_EXTERNS \/AL \/Zp4\n";
  print OUT "set LINK=/CO /MAP:FULL /SE:1024 /nod /noe /noi";
  print OUT "cl -c -Fo$outfile.obj -I$ENV{SRCTOP}\\test\\fvt\\inc -I. -Tc
$file.c\n";
  print OUT "link \@1.rsp;\n";
  close(OUT);
  &make_rexx_file("buildos.cmd","buildos.bat");
  &verbose_system("buildos.cmd");
  &rm("buildos.bat");
  &rm("buildos.cmd");
 }
}


##Internal
#cobolcompile
#

sub  cobolcompile
{
  local( $file ) = @_;

  if ($OS2)
  {
    &verbose_system( "cobol $file.cbl /NOTRUNC /NOQUERY;" );
  }
  if($AIX)
  {
    &verbose_system("cob -cx $file.cbl");
  }
}


##Internal
#cppcompile
#

sub cppcompile
{
  ( $file, $outfile ) = @_;

  if (!$outfile) { $outfile = $file; }

  # this is only used by OS/2 to compile jrouter so far

  if ($AIX)
  {
  }
  if ($OS2)
  {
    &verbose_system( "icc /C /Fd /Fi /Ft- /Gd /Ge+ /Gm /Gn /Si /Ti /Wcnv- /
```

```
G4 /D_cpp /DS_DEBUG $file.cpp" );
   }
   if ($WINT)
   {
   }
}


##Internal
#finderror
#
# Find whether the given file contains error messages.
# If a second file is specified, output to it the errors found.

sub finderror
{
    local($infile,$outfile) = @_;
    local($error);
    $infile || die "finderror needs an input file!\n";
    open(FINDERRORIN,$infile) || die "finderror cant open file:$infile\n";
    if($outfile){
        open(FINDERROROUT,">$outfile") || die "finderror cant open
file:$outfile\n";
    }
    $error = 0;
    while(<FINDERRORIN>){
        if(/error/i || /SYS[0-9]+/
           || /Not connected/i
           || /continue with mget/i
           || /is not recognized/i
           || /The system cannot/i
           || /not a plain file/i
           || /is read-only/i
           || /cannot be found/i
           || /The process cannot/i
           || /cannot find/i
           || /No such file or directory/
           || /denied/i
           || /Sharing violation/){
            unless(/ 0 errors/ || /error IM2601/
                   || /No errors detected/
                   || /complete with no errors/ || /senderror/i){
               $error = 1;
               if($outfile){
                   print FINDERROROUT $_;
               } else {
                   last;
               }
            }
        }
    }
    close(FINDERRORIN);
    close(FINDERROROUT);
    return $error;
}


##Internal
#fmtcompile
#

sub  fmtcompile
{
   local( $file ) = @_;

   if ($OS2)
   {
      system( "fmt2sqf SQLOS2 $file > $file.for" );
   }
   if ($AIX)
   {
      system( "fmt2sqf SQLAIX $file > $file.f" );
   }
   &fortrancompile( $file );
}


##Internal
#fortrancompile
#

sub  fortrancompile
{
   local( $file ) = @_;
   local($fortran);

   $fopts = &sl($fopts);

   if ($OS2)
   {
      system( "wfc386 /NOR /d2 /debug $file.for /FO=$file.obj" );
   }
   if ($AIX)
   {
      if($REAL_AIX)
      {
         $fortran = "xlf";
      } else {
         $fortran = "f77";
      }
      system( "$fortran -w -c $file.f $fopts -I$ENV{HOME}/sqllib/include -I/
wsdb/test/fvt/inc -I$ENV{HOME}/sqllib/include" );
   }
}


##Internal
#getincdir
#
# Return the absolute path of the relevant DB2 lib's (ie SRCTOP)
# - this function returns the absolute path of the top of the
#   build tree (Unix only)
# - this function should be used for specifying include and lib paths
#   for tools which use Db2 libraries

sub getincdir
{
  local($cdir, %sv);

  unless ($ENV{'INSTTOP'})
  {
    if ($AIX)
    {
      $cdir = $ENV{PWD};

      # attempt to traverse the tree downwards to find the INST dir
      getD: while ($cdir)
      {
        if (-d "$cdir/INST") {last getD;}
        $cdir = substr($cdir,0,rindex($cdir,"/"));
      }

      # remove the NNN's from the leading "wsdbNNN" path
      $cdir =~ s@^/wsdb\d+@/wsdb@;

      # if we don't find anything, use BLDPATH as the default
      if ($cdir eq "")
      {
        %sv = &get_sysval;
        $cdir = $sv{'bldpath'};
      }

      $ENV{'INSTTOP'} = $cdir;
    }
  }

  return $ENV{'INSTTOP'};
}


##Internal
#lib
#

sub lib
{
    local($outfile, $files) = @_;


    $files = &sl4(&sl($files));
    $outfile = &sl($outfile);

    if ($AIX)
    {
```

```
      $files =~ s/(\S+)/\1.o/g;
      &rm("$oufile.a");
      system("ar rv $outfile.a $files");
   }
   if ($OS2)
   {
      $outfile = &sl4($outfile);
      $files =~ s/^\s+//g;
      $files =~ s/\s+$//g;
      $files =~ s/\s+/\-\+/g;
      system( "lib $outfile-+$files;" );
   }
   if ($WINT)
   {
      $files =~ s/(\S+)/\1.obj/g;
      system("LIB /DEBUGTYPE:BOTH /OUT:$outfile.lib $files");
   }
   if ($WIN)
   {
      $outfile = &sl4($outfile);
      open(OUT, ">buildos.bat") || warn "Can't open buildos.bat";
      print OUT "set PATH=t:\\msvc\\bin;%PATH%\n";
      print OUT "set
INCLUDE=t:\\dos\\db2_v3\\include;t:\\dos\\tcpdos\\include\\rsa;.;t:\\msvc\\inc
lude;%INCLUDE%\n";
      print OUT "set LIB=t:\\dos\\tcpdos\\lib;t:\\msvc\\lib;%LIB%\n";
      for (split(/ /,$files))
      {
        print OUT "lib $outfile-+$_\;\n";
      }
      close(OUT);
      &make_rexx_file("buildos.cmd","buildos.bat");
      system("buildos.cmd");
      &rm("buildos.bat");
      &rm("buildos.cmd");
   }
}


##Internal
#llink
#

sub llink
{
   local($outfile, $files) = @_;
   local( $reallinkopt );


   $files = &sl4(&sl($files));
   $outfile = &sl4(&sl($outfile));
   $reallinkopt = &sl4(&sl($linkopt));

   if ( $AIX )
   {
      $files =~ s/(\S+)/\1.o/g;
      &verbose_system( "cc -o $outfile$linkext $files $reallinkopt" );
   }
   if ( $OS2 )
   {
      $files =~ s/(\S+)/\1.obj/g;
      &verbose_system("icc /Tdp /DE /Fe$outfile$linkext.exe $reallinkopt
$files OS2386.LIB TCP32DLL.LIB SO32DLL.LIB $outfile$linkext.def ");
   }
   if ( $WINT )
   {
      $files =~ s/(\S+)/\1.obj/g;
      &verbose_system("link /DEBUG /DEBUGTYPE:both /
OUT:$outfile$linkext.exe $reallinkopt $files WSOCK32.LIB
KERNEL32.LIB");
   }
   if ($WIN)
   {
     open(OUT, ">buildos.bat") || warn "Can't open buildos.bat";
     print OUT "set PATH=t:\\msvc\\bin;%PATH%\n";
     print OUT "set
INCLUDE=t:\\dos\\db2_v3\\include;t:\\dos\\tcpdos\\include\\rsa;.;t:\\msvc\\inc
lude;%INCLUDE%\n";
     print OUT "set LIB=t:\\dos\\tcpdos\\lib;t:\\msvc\\lib;%LIB%\n";
     print OUT "link $files,$outfile.exe,nul,\/NOD \/NOE llibce oldnames
socketl \/stack:32000\;\n";
     close(OUT);
     &make_rexx_file("buildos.cmd","buildos.bat");
     &verbose_system("buildos.cmd");
     &rm("buildos.bat");
     &rm("buildos.cmd");
   }

}


##Internal
#make_rexx_file
#

sub make_rexx_file {
  local( $rexxfile, $batfile ) = @_;
  if ($WIN)
  {
    open(OUT, ">$rexxfile") || warn "Can't open $rexxfile";
    print OUT "\/* This calls a DOS session modified for MSVC1.5*\/\n";
    print OUT "parse arg szCommandComArgs .\n";
    print OUT "parse source . . szRexxFileName .\n";
    print OUT "select\n";
    print OUT "when 'CMD' = address()  then do\n";
    print OUT "'t:\\wintools\\STARTDOS' szRexxFileName
szCommandComArgs\n";
    print OUT "end\n";
    print OUT "when 'STARTDOS' = address()  then do\n";
    print OUT "rc = SetCommandArgs( \"\/c $batfile\" szCommandComArgs )
\n";
    print OUT "rc = StartWindowed()\n";
    print OUT "rc = AddDosSetting( \"DPMI_DOS_API=ENABLED\" )\n";
    print OUT "rc = AddDosSetting( \"DPMI_MEMORY_LIMIT=10\" )\n";
    print OUT "rc = StartBackground( )\n";
    print OUT "rc = ExecSynchronous( )\n";
    print OUT "end\n";
    print OUT "end\n";
    close(OUT);
  }
}


##Heading
#Jrouter and regression


##External
#Jrouter Information
#
#  For information on jrouter, jclient and associated commands,
#  please refer to the specific jrouter entry on the regression homepage.
#


##Internal
#fvt_checkin
#

sub  fvt_checkin
{
  local( $file, $local, $comment ) = @_;


  if ($local)
  {
    return;
  }

  if ($REAL_AIX)
  {
#     &quiet_system( "sccstool checkin $repfile $comment" );
     while( 1 )
     {
        $return = &quiet_system_return( "sccstool checkin $file $comment" );
        printf( "$return\n" );
        if ($return =~ /checkin successful/ )
        {
          return;
        }
        print "Waiting for checking of $file ...\n";
        sleep(20);
     }
  }
```

```
}


##Internal
#fvt_checkout
#

sub  fvt_checkout
{
  local( $file, $local, $comment ) = @_;
  local( $rc );


  if ($local)
  {
    return;
  }

  if ($REAL_AIX)
  {
    while( 1 )
    {
      $return = &quiet_system_return( "sccstool checkout $file $comment" );
      printf( "$return\n" );
      if ($return =~ /checkout successful/ )
      {
        return;
      }
      print "Waiting for report $file to be available for editing ...\n";
      sleep(20);
    }
  }
}


##Internal
#fvt_command_2_bucket
#
# takes in a regr command and parses out the full bucket name
#
sub fvt_command_2_bucket
{
  local( $bucket ) = shift;
  local( $args ) = shift;

  local( $class ) = "";
  local( $type ) = "";
  local( $fvt_class, $tempn, $tempp, $temps );


  # go through the command, build up the original class and type variables

  ( $tempn ) = ( $args =~ /\/nodes:(\d+)/ );
  ( $tempp ) = ( $args =~ /\/nameparm:(\S+)/ );
  ( $temps ) = ( $args =~ /\/smpdeg:(\d+)/ );
  if ($tempn)
  {
    $class .= "n($tempn) ";
  }
  if ($tempp)
  {
    $class .= "p($tempp) ";
  }
  if ($temps)
  {
    $class .= "s($temps) ";
  }

  if ($args =~ /\@dbcs/)
  {
    # get the dbcs language from the suffix of the directory - no suffix is
japanese

    ( $type ) = ($args =~ /\@dbcs\.\S\S_(\S+)\// );
    if (!$type) { $type = "JP"; }
    $type = "D-$type";
  }

  if ($args =~ /\@cs/)
  {
    # this is real hokey
    # find out the job class of this job
    # if regr - it is running loopback for this platform

    $type = "CS-";
    $fvt_class = $ENV{FVT_JOB_CLASS};
    $fvt_class =~ s/ //g;
    $class .= " $fvt_class";

    if ($fvt_class eq "regr")
    {
      $type .= &get_blduname( );
    }
    else
    {
      &fvt_read_servers( );
      $type .= $fvt_serverplats{$fvt_class};
    }
  }

  # call the common routine to build the bucket name

  $bucket = &build_bucket_name( $bucket, $type, $class );

  return $bucket;
}


##Internal
#fvt_get_platforms
#
#   returns a list of platforms currently supported
#

sub fvt_get_platforms

{
  local( $plat ) = @_;
  local( @plats );

  foreach $plat ( @slaveposn )
  {

  }

  @plats = ( 'aix', 'os2', 'wint', 'win95', 'mvs', 'hp', 'sun' );

  return @plats;

}


##Internal
#fvt_open_slaves
#
# opens the slaves.lst file and caches the information into an internal array
# fvt_slave_list.  This array can be used by any internal macro after first
calling
# fvt_open_slaves to ensure the array is set up
#

sub fvt_open_slaves

{
  local( $F, $slave );


  if ($fvt_slave_list_read)
  {
    return;
  }

  &get_sysval( );
  $F = "$sysval{bldpath}/test/slaves.lst";

  open(IN,$F) || die "Unable to open $F";
  while (<IN>)
  {
    chop;
    next if substr( $_, 0, 1 ) eq "*";
    $slave = substr( $_, 0, 14 );
    $slave =~ s/ //g;

    unshift( @fvt_slave_list, $slave );
```

```
    @fvt_slave_list_info{$slave} = $_;
  }
  close(IN);

  $fvt_slave_list_read = 1;

}


##Internal
#fvt_plat_2_suffix
#

sub fvt_plat_2_suffix
{
  local( $plat ) = @_;
  local( $suffix );


  $plat =~ tr/[A-Z]/[a-z]/;
  $suffix = ".$plat";
  if ($suffix eq ".aix")
  {
    $suffix = "";
  }

  return $suffix;

}


##Internal
#fvt_read_servers
#
# creates 3 arrays - fvt_servers, fvt_serverids, fvt_serverplats from the info in
# the slaves.lst file.
#

sub fvt_read_servers

{
  local( $class, $slave, $server, $id );


  &fvt_open_slaves( );

  foreach $slave (@fvt_slave_list)
  {
    if (&fvt_slave_is_server( $slave ))
    {
      $class = substr( @fvt_slave_list_info{$slave}, $slaveposn{srvclass},
14 );
      $class =~ s/ //g;
      $server = substr( @fvt_slave_list_info{$slave}, $slaveposn{id}, 19 );
      $server =~ s/ //g;
      ( $id, $server ) = split( '@', $server );
      $fvt_servers{$class} = $server;
      $fvt_serverids{$class} = $id;
      $fvt_serverplats{$class} = &fvt_slave_platform( $slave );

#       printf( "slave $slave is a server for class $class, server $server, id $id,
plat $fvt_serverplats{$class}\n" );
    }
  }
}


##Internal
#fvt_slave_abbr
#
#   returns the two character abbreviation for this slave
#

sub fvt_slave_abbr

{
  local( $slave ) = @_;
  local( $abbr );

  &fvt_open_slaves( );

  $abbr = substr( @fvt_slave_list_info{$slave}, $slaveposn{abbr}, 2 );
  $abbr =~ s/ //g;
  if ($abbr eq "" )
  {
    $abbr = substr( $slave, 0, 1 );
    $abbr =~ tr/[a-z]/[A-Z]/;
    $abbr = "x$abbr";
  }
  else
  {
    $abbr =~ tr/[a-z]/[A-Z]/;
  }

  return $abbr;

}


##Internal
#fvt_slave_id
#
#   returns the actual id for this slave, values are returned as a list -
#   id, machine
#

sub fvt_slave_id

{
  local( $slave ) = @_;
  local( $id, @pair );

  &fvt_open_slaves( );

  $id = substr( $fvt_slave_list_info{$slave}, $slaveposn{id}, 19 );
  $id =~ s/ //g;
  ( @pair ) = split( '@', $id );

  return @pair;

}


##Internal
#fvt_slave_is_downlevel
#
#   returns 1 if the indicated slave is downlevel
#

sub fvt_slave_is_downlevel

{
  local( $slave ) = @_;

  &fvt_open_slaves( );

  if (substr( $fvt_slave_list_info{$slave}, $slaveposn{downlv}, 1 ) eq "x")
  {
    return 1;
  }

  return 0;

}


##Internal
#fvt_slave_is_mpp
#
#   returns 1 if the indicated slave is a mpp slave
#

sub fvt_slave_is_mpp

{
  local( $slave ) = @_;

  &fvt_open_slaves( );

  if (substr( $fvt_slave_list_info{$slave}, $slaveposn{mpp}, 1 ) eq "x")
  {
    return 1;
  }
```

```
  return 0;

}


##Internal
#fvt_slave_is_server
#
#   returns 1 if the indicated slave is a server
#

sub fvt_slave_is_server

{
  local( $slave ) = @_;


  &fvt_open_slaves( );

  if (substr( $fvt_slave_list_info{$slave}, $slaveposn{server}, 1 ) eq "x")
  {
    return 1;
  }

  return 0;

}


##Internal
#fvt_slave_platform
#
#   returns the platform of the indicated slave
#

sub fvt_slave_platform

{
  local( $slave ) = @_;
  local( $info, $plat, @plats );

  &fvt_open_slaves( );
  @plats = &fvt_get_platforms( );

  $info = @fvt_slave_list_info{$slave};
  foreach $plat ( @plats )
  {
    if (substr( $info, $slaveposn{$plat}, 1 ) eq "x")
    {
      return $plat;
    }
  }

  return "unknown";

}


##Internal
#get_jclient_parms
#
# take @args and %opts from get_parms and look for jclient specific
# options, eg. /os:, /release:
#
# translate these into a form consumable by jclient

sub get_jclient_parms
{
  local( $jclient_opts ) = "";


  &get_parms( );

  if ($opts{"os"})
  {
    $jclient_opts = $jclient_opts." -q".$opts{"os"};
  }
  if ($opts{"release"} eq "")
  {
    $opts{"release"} = &get_release;
  }
  if ($opts{"release"} eq $alt_release )
  {
    $jclient_opts = $jclient_opts." -a";
  }

  # for PE v1.2 - translate /release:db2_v1 into /os:pe

  if ($opts{"release"} eq "db2pe_v12")
  {
    $jclient_opts = " -qpe";
  }

  return( $jclient_opts );
}


##internal
#get_jclient_release
#
# take in a flag specifying alternate or main release,
# return the text string corresponding to that release
#

sub get_jclient_release
{
  local( $alternate ) = @_;


  if ($alternate)
  {
    return $alt_release;
  }
  else
  {
    return $main_release;
  }

}


##Internal
#get_slaves
#

sub get_slaves

{
  local( $type ) = @_;
  local( @slaves );


  &fvt_open_slaves( );

  $slavepos = $slaveposn{ $type };
  foreach (%fvt_slave_list_info)
  {
    if (substr( $_, $slavepos, 1 ) eq 'x' )
    {
      chop( $slave = $_ );
      $slave =~ s/\s.*//;
      push( @slaves, $slave );
    }
  }

  return( @slaves );
}


##Internal
#simple_jclient
#

sub simple_jclient
{
  local( $jclient_cmd ) = @_;
  local( $jclient_args ) = &get_jclient_parms( );

  &quiet_system( "jclient -i$jclient_cmd $jclient_args" );
}


##Internal
#trans_slave_uid
```

```
#

sub trans_slave_uid

{
  local( $ckslave, $release ) = @_;
  local( $slaveid, $slavelong );


  &get_sysval( );
  $F = "$sysval{bldpath}";
  if ($release ne "" && $release ne $main_release)
  {
    # this is not the main release - must get to bldpath on a previous release
    if ($AIX)
    {
      $F = "/wsdb/$release/daily"
    }
    else
    {
      $F = "u:/bldtree/$release/nightly";
    }
  }
  $F .= "/test/slaves.lst";

  open(IN,$F) || die "Unable to open $F";
  while (<IN>)
  {
    $temp = substr( $_, 0, 14 );
    $temp =~ s/ //g;
    if ($temp eq $ckslave)
    {
      $slavelong = substr( $_, 42 );
      ( $slaveid ) = ( $slavelong =~ /(\S+)\@/ );
    }
  }
  close(IN);

  if (!$slavelong)
  {
    die "can not locate slave $ckslave\n";
  }

  return( $slaveid );
}


##External
#Package fvt_nodes_cfg
#
# The fvt_nodes_cfg package is used to access and change the MPP node
# configuration data (on UNIX found in db2nodes.cfg).
#
# package fvt_nodes_cfg;
#          &init   ( $savename [, $mode | $fromsavename]     );
# @data    = &list   ( [$savename, @namelist [, @attributes ]] );
#          &use    ( $savename, @namelist                    );
# @namelist = &resolve( $savename, @namelist [, $qualifier]     );
# @lines   = &dump   ( $savename [, @nodelist]                 );
# $value   = &get    ( $savename, $variable                    );
#          &set    ( $savename, $variable, $value            );
#          &clear  ( $savename [, $variable]                 );
#
# A note on style:  all variable and function names mentioned in the text will
# be pre- and post-fixed by == (two equal signs);  code examples, however,
can
# be copied without changes (and will be written to be used in the default
# package, package ==main==).
#
#
# A NOTE ON PACKAGES
# If you have never used packages before, here is a brief tutorial.
#
#   +-----------------------------------------------------------------+
#   | For package fvt_nodes_cfg element | Use                         |
#   +-----------------------------------------------------------------+
#   |  function &list                   | &fvt_nodes_cfg'list         |
#   |  variable $STD_NAMES              | $fvt_nodes_cfg'STD_NAMES    |
#   +-----------------------------------------------------------------+
#
# For the font impaired, that is a single quote after the package name.
#
#
# THE SOLUTIONS TO MOST PROBLEMS
# In most cases, all that is needed is a list of the node numbers that are in
# use.  If this is all the functionality you need, simply call ==&list==.  Pass
# it no parameters.  It will return a list of node numbers that are currently
# in use.
#
# Example:
#    @nodelist = &fvt_nodes_cfg'list;
#
# A slightly more flexible version of ==&list== allows you to retrieve other
# details about the node configuration without going to the trouble of naming
# nodes.  Call this extended version of ==&list== with ==undef== as the first
# parameter, and a list of one or more of NUMBERS, HOSTS, PORTS, and
NETWORKS
# to receive a set of data about each node in the current configuration.
#
# Example:
#    @data = &fvt_nodes_cfg'list( undef, "NUMBERS", "HOSTS" );
#       # @data contains ( line 1 node number, line 1 host,
#       #                  line 2 node number, line 2 host, ... );
#
#
# For other common "one-step" type operations (like reordering from 0 the
node
# numbers in the configuration) check out the ==fvt_nodes_cfg_macros==
package.
#
#
# THE SOLUTION TO LEAST PROBLEMS
# For all other purposes, you will need to understand the node naming systems
# used by the package.  There are currently two naming systems, standard
names
# and mln names.  And now for a useful tidbit from your sponsor: when
assigning
# node names, the package is careful to ensure that "lower" names are always
# assigned to lower port numbers.  This means, it is (almost) always safe to
# use the first X nodes, but seldom safe to use the last X nodes.
#
#
# THE SIMPLE WAY: STANDARD NAMES
# The standard naming convention is very simple.  The catalog node is
assigned
# the name "c".  All other nodes are named, "A", "B", "C", ..., "Z", "AA",
# "AB", ....
#
#
# THE FLEXIBLE WAY:MLN NAMES
# The mln naming convention provides more flexibility (and requires more
work)
# than standard names.  Unlike in the standard naming convention, the hosts
are
# assigned the names ("c" to the catalog host, then "A", "B", ...).  For each
# host, each port number (in ascending order) is then assigned a unique
"name"
# (a number counted contiguously from 0).  The node name is then made from
the
# host name by prepending an "h", and appending the port name (unless the
port
# name is 0). The node name "c" points to the actual catalog node, which
could
# be any of the "hc" names.  I think an example will make all clear.
#
#   +-----------------------------------------------------------------+
#   | Name(s) | Number | Host    | Port | Special notes                |
#   +-----------------------------------------------------------------+
#   | hc2, c |   10   | sirius |   6  | Catalog node.  Notice the port. |
#   | hA     |   20   | vega   |   0  |                                |
#   | hB     |   30   | polaris |  0  |                                |
#   | hc     |   40   | sirius |   0  | The catalog host.              |
#   | hc1    |   50   | sirius |   2  | Notice the names/ports on sirius |
#   | hB1    |   60   | polaris |  1  |                                |
#   | hC     |   70   | sol    |   0  | hC, not hc                     |
#   +-----------------------------------------------------------------+
#
#
# A FRAGILE ENVIRONMENT
# The data the package needs to do its work (at least the part that isn't read
# directly out of the node congifuration data) is stored in named sets of
# environment variables (we shall call these named sets "saves").  Except in
# certain cases (listed below) these environment variables should not be
# touched, or even examined.  For the sake of avoiding collisions, all
```

```
# environment variables used by this package begin with the characters
# "FVT_NODES_CFG".
#
#
# GETTING STARTED: &init
# Before being able to use node names to manipulate the configuration, the
# package must assign them, and create a "save" to reference them with. The
# ==&init== function is used to do this. It basically grabs a snapshot of the
# node configuration at the time it is called, and sets up the save
# accordingly. You can tell it to make new standard names, new mln names, or
# to assign the names based on the names from an existing save.
#
# Here are some examples (refer to the above node configuration):
#    # create set ORIGINAL with standard names
#    &fvt_nodes_cfg'init( "ORIGINAL", $fvt_nodes_cfg'STD_NAMES );
#
#    # create set MLN with mln names
#    &fvt_nodes_cfg'init( "MLN", $fvt_nodes_cfg'MLN_NAMES );
#
#    # create NEWMLN copy of MLN based on MLN's names
#    # you would usually only do this after changing the node configuration
#    &fvt_nodes_cfg'init( "NEWMLN", "MLN" );
#
#
# THE POINT OF LEAST: &use
# If you are bothering to name your nodes (HEY! get your mind out of the
# gutter!), its probably because you want to be able to change the node
# configuration. You supply ==&use== with the name of a save, and a list of
# node names, and it will rewrite the node configuration to match your request.
# For those who need it, you can control the presence (and, to a point, the
# value) of the network name for each node. You can also override the normal
# node number. Each node name argument takes the form
# "nodename:networkcontrol:nodenumber". Each colon (and everything that
# follows it) is optional. The following chart shows the possible values of
# networkcontrol (if it is supplied at all):
#
#   +-------------------------------------------------------------+
#   | networkcontrol | Meaning                                    |
#   +-------------------------------------------------------------+
#   | unset, "", n   | Use the network name the node had when ==&init==   |
#   |      x         | Do not use a network name                  |
#   |      h         | Use the host name as the network name      |
#   +-------------------------------------------------------------+
#
# Here are some examples (refer to the ==&init== examples above):
#    # 1. Use three nodes from save MLN, the last one with no network name
#    &fvt_nodes_cfg'use( "MLN", "c", "hB", "hB1:x" );
#
#    # 2. Use only the catalog node, but change its number to 300
#    &fvt_nodes_cfg'use( "MLN", "c::300" );
#
#    # 3. Use all the nodes.
#    #    NOTE: you cannot supply networkcontrol or number overrides with *
#    &fvt_nodes_cfg'use( "MLN", "*" );
#
#    # 4. Use three nodes from ORIGINAL
#    &fvt_nodes_cfg'use( "ORIGINAL", "c", "A", "B" );
#
#
# REVISIONIST HISTORY: &list
# If you are using node names, ==&list== can do lots of extra things for you.
# In addition to being able to return node numbers, you can also get a list of
# hosts, ports, network names, and node names. ==&list== always looks up
your
# requests in the actual node configuration, so it is not useful for getting
# information about nodes not currently in use. You pass ==&list== a save
# name, a list of node names (or "*", for all nodes), and an optional list of
# attributes you wish to have returned (NUMBERS, HOSTS, PORTS,
NETWORKS,
# NAMES). If you do not supply any attributes, ==&list== returns the node
# numbers.
#
# NOTE: You will get fewer items returned to you if any of the names you
# ==&list== are not actually in the node configuration.
#
# Examples (example numbers refer to the ==&use== examples above):
#    # After example 3
#    @data = &fvt_nodes_cfg'list( "MLN", "*" );
#          # @data is ( 10, 20, 30, 40, 50, 60, 70 )
#
#    # After ==&use== example 1
#    @data = &fvt_nodes_cfg'list( "MLN", "hB", "hB1", "NAMES",
"PORTS" );
#          # @data is ( "hB", 0, "hB1", 1 )
#
#    # After ==&use== example 4
#    @data = &fvt_nodes_cfg'list( "ORIGINAL", "c", "D", "A", "NAMES" );
#          # @data is ( "c", "A" );
#
#
# MAKING THE MOST OF THE LEAST: &resolve
# Sometimes you need a special set of nodes. Like all the nodes but the
# catalog. Or the first 3 nodes. Or the last three nodes. ==&resolve== is
# used to figure out these things. You give it a save name, a list of node
# names (or a "*" for all nodes), and an optional subset parameter, and it will
# return to you the correct list of node names.
#
# Here are some examples:
#    # A list of all nodes
#    @names = &fvt_nodes_cfg'resolve( "ORIGINAL", "*" );
#
#    # A list of all non-catalog nodes
#    @names = &fvt_nodes_cfg'resolve( "MLN", "*", "-c" );
#
#    # Another list of all non-catalog nodes
#    @names = &fvt_nodes_cfg'resolve( "ORIGINAL", "A", "B", "c", "-c" );
#
#    # The first three non-catalog nodes
#    # NOTE: the plus sign (+) IS REQUIRED
#    @names = &fvt_nodes_cfg'resolve( "MLN", "*", "-c" );
#    @names = &fvt_nodes_cfg'resolve( "MLN", @names, "+3" );
#
#    # The last three non-catalog nodes
#    @names = &fvt_nodes_cfg'resolve( "ORIGINAL", "*", "-c" );
#    @names = &fvt_nodes_cfg'resolve( "ORIGINAL, @names, "-3" );
#
#
# FOR POSTERITY: &dump
# Sometimes, you need to print out the current node configuration.
==&dump==
# is used to do that. You pass it a save name and an optional list of node
# names, and it returns a list of formatted strings, suitable for printing
# (you, however, have to supply the newlines, if needed). If you do not supply
# a list of node names, the memory configuration is dumped. If you do supply
a
# list of node names, the actual node configuration for those names is dumped.
#
# Examples:
#    @lines = &fvt_nodes_cfg'dump( "MLN", "c", "hB1" );
#    $" = "\n"; print "@lines\n"; $" = " ";
#
#
# PEEKING AND POKING: &get, &set, &clear, public save variables
# There are a number save variables you can set to modify the behaviour of the
# package (specifically ==&use==), when operating on that save. In addition,
# there are several functions supplied to manipulate these variables.
#
# ==&get== returns the value of the specified save variable
#    $ordering = &get( "ORIGINAL", "ORDERING" );
#
# ==&set== sets the value of a specified save variable
#    &set( "MLN", "ORDERING", $ordering );
#
# ==&clear== deletes a specified save variable
#    &clear( "MLN", "ORDERING" );
#
# ==&clear== can also delete an entire save
#    &clear( "MLN" );
#
# The following chart shows all the save variables, and what each is used for.
#
#   +-------------------------------------------------------------+
#   | Variable        | Values        | Meaning                    |
#   +-------------------------------------------------------------+
#   | ORDERED         | unset         | Use random ordering        |
#   |                 | "$start $step" | $step can be negative (or even 0)|
#   |                 |               |                            |
#   | NO_SORT         | unset         | Ensure db2nodes.cfg node numbers |
#   |                 |               |    ascend                  |
#   |                 | set           | Don't sort db2nodes.cfg    |
#   |                 |               |                            |
#   | NEW_NODE_NUMBERS | unset         | For random ordering, use node
|
#   |                 |               |    numbers found at ==&init==   |
```

```
#   |                | set            | For random ordering, new node    |
#   |                |                |    numbers will be created       |
#   +-----------------------------------------------------------------+
#
#
# KNOWN LIMITATIONS AND BUGS
# There are a few known limitations at this time:
# 1) Both standard and mln names identify nodes by the same method -- host
name
#    and port number.  It would be more intuitive for standard names to rely on
#    the node number.
# 2) If you use the ==dbstart restart== functionality to move nodes around, the
#    standard names setup will fail (unless you have reset the nodes before
#    calling any of the package functions).  See limitation 1 for why.  If you
#    need to move nodes around with ==db2start restart==, use mln names.


##Internal
#Package fvt_nodes_cfg
#
# SPECIAL VARIABLES FOR SPECIAL PEOPLE
# Occasionally, there will be a bucket that needs to break some rules.  Like
# the ability to create logical nodes it wasn't supplied with.  Here are some
# extra save variables that should only be used by special people.  Everyone
# else should leave them alone (ie. not even know about them).
#
#   +-----------------------------------------------------------------+
#   | Variable        | Use                                           |
#   +-----------------------------------------------------------------+
#   | FREEDOM         | If this is set, and mln names are being used,     |
#   |                 | the port name portion of the mln name can be used |
#   |                 | as a port number, and new logical nodes can be    |
#   |                 | created.  NOTE: this will only work if the        |
#   |                 | originally supplied ports are numbered            |
#   |                 | consecutively from 0.                             |
#   |                 | Example:                                          |
#   |                 |    # TEST contains hc, hc1, hc2 (ports 0, 1, 2)   |
#   |                 |    &fvt_nodes_cfg'set( "TEST", "FREEDOM" );       |
#   |                 |    &fvt_nodes_cfg'use( "TEST", "hc", "hc4" );     |
#   |                 |    # config now has ports 0, 4                    |
#   |                 |                                                   |
#   | GARBAGE_NETWORK | If the networkname token passed to
==&use== is    |
#   |                 | "g", the contents of this variable will be used as|
#   |                 | the network name.  If this variable is not set,   |
#   |                 | "g" will do nothing...                            |
#   +-----------------------------------------------------------------+
#
#
# THE INTERNALS
# Well, the code is reasonably well documented internally, so I will only
# describe the internal save variables and the remaining functions here.
#
# @numlist = &nodenums    ( $savename, @namelist );
# @numlist = &makenodenums( $savename, $count    );
# @data    = &slurp       ( @fields              );
# @data    = &name        ( $savename, @fields   );
# @lines   = &sortcfg     ( @lines               );
# $cmp     = &numerically ( $a, $b               );
#
#
# THE VARIABLES
# The first things to describe are the internal save variables.
#   +-----------------------------------------------------------------+
#   | Variable        | Use                                           |
#   +-----------------------------------------------------------------+
#   | INIT            | Set on entrance to ==&init==, cleared on exit.    |
#   |                 | Can be used to let special things happen in other|
#   |                 | functions during initialization.                  |
#   |                 |                                                   |
#   | NAMING_CONVENTION | Set to $MLN_NAMES or $STD_NAMES by
==&init==     |
#   |                 |                                                   |
#   | NODES           | This is the big one.  Contains a space separated |
#   |                 | list of token pairs.  Each pair is a node name,   |
#   |                 | and a node information packet in the format       |
#   |                 | $host:$port:$netUsed, where $netUsed is $TRUE if |
#   |                 | a network name was supplied, $FALSE otherwise.    |
#   |                 | Example:                                          |
#   |                 |   "c host1:0:1 A host1:1:0 B host1:2:0"           |
#   |                 |                                                   |
#   | NODE_NAMES      | This is similar to NODES, except the each pair is|
#   |                 | reversed, and the $netUsed data is missing.       |
#   |                 | Example:                                          |
#   |                 |   "host1:0 c host1:1 A host1:2 B"                 |
#   |                 |                                                   |
#   | CATALOG_LINK    | Contains the name of the catalog node.            |
#   |                 | In $STD_NAMES, always "c".  In $MLN_NAMES, "hc?" |
#   |                 |                                                   |
#   | NETWORKS        | Contains a space separated list of token pairs.  |
#   |                 | Each pair is a node name, and a network name.  If|
#   |                 | a host has no network name, it won't be in the    |
#   |                 | list.                                             |
#   |                 |                                                   |
#   | NUMBERING       | Holds all active sets of node numbers.  See       |
#   |                 | &nodenums for details.                            |
#   |                 | Example:                                          |
#   |                 |   "RANDOM\n" .                                    |
#   |                 |   "hc 10 hB 39 hB1 94\n" .                        |
#   |                 |   "ORDERED 10 1\n" .                              |
#   |                 |   "hc 10 hB 11 hB1 12\n" .                        |
#   |                 |   "FREEDOM RANDOM\n" .                            |
#   |                 |   "hc 22 hc1 399 hc2 783 hc3 903\n"               |
#   +-----------------------------------------------------------------+
#
#
# NUMBERING SYSTEMS: &nodenums, &makenodenums
# ==&nodenums== and ==&makenodenums== go together.
==&nodenums== is used to
# retrieve a set of node numbers for a particular set of node names, under the
# current setup (ORDERED, FREEDOM, NEW_NODE_NUMBERS, etc.) of
the specified
# save name.  If ==&nodenums== can't find a saved set of node numbers for
the
# current setup, it calls &makenodenums to generate one.
#
#
# PARSING AND LABELLING: &name, &slurp
# ==&name== and ==&slurp== also go together.  Both functions take a list of
# field identifiers, and return a list of values for each field, for each node
# in the current db2nodes.cfg file.  Basically, ==&name== passes the list of
# fields it receives on to ==&slurp==, which parses the current db2nodes.cfg
# file, and returns the appropriate set of values.  ==&name== then either looks
# up or assigns names (depending on whether the naming operation has
already
# been done), and inserts the correct node name at the beginning of each data
# set returned to it from ==&slurp==.  It then returns the new list.
#
# The following are the fields you can request from these two functions.
#   +-----------------------------------------------------------------+
#   | Field | What gets put on the list                                |
#   +-----------------------------------------------------------------+
#   |   #   | The node number                                          |
#   |   H   | The host name                                            |
#   |   P   | The port number                                          |
#   |   N   | The network name                                         |
#   |   n   | $TRUE if a network name is present, else $FALSE          |
#   |   _   | The empty string (its a placeholder/noop kind of thing)  |
#   +-----------------------------------------------------------------+
#
# Using the following node configuration:
#    1 host1 0 host1_net
#    2 host2 0
#    3 host1 1
#    4 host2 1 host2_net
#    5 host3 0
# then running:
#    &fvt_nodes_cfg'init( "SAVE", $fvt_nodes_cfg'STD_NAMES );
#    &fvt_nodes_cfg'set( "SAVE", "ORDERED", "1 10" );
#    &fvt_nodes_cfg'use( "SAVE", "D", "C", "B", "A", "c" );
# example:
#    @data = &fvt_nodes_cfg'slurp( "#", "H" );
#          # data is ( 1, "host3", 11, "host2", 21,
#          #           "host1", 31, "host2", 41, "host1" )
#
#    @data = &fvt_nodes_cfg'name( "SAVE", "N", "n" );
#          # data is ( "D", "", 0, "C", "host2_net", 1, "B", "", 0,
#          #           "A", "", 0, "c", "host1_net", 1 );
#
#
# UTILITIES: &sortcfg, &numerically
# ==&sortcfg== is used to sort the db2nodes.cfg data before it is returned from
# ==&slurp==.  The sort ==&sortcfg== performs makes sure that, within each
host
```

```
# name, the port numbers are always in ascending order.  This is done to
ensure
# that it is (almost) always acceptable to used the first X node names (it is
# illegal to have port x>0 in the configuration if port 0 isn't in the
# configuration).
#
# ==&numerically== is not really a function, put a sort routine (for use with
# the perl sort command).  It is used for numeric sorting of data (specifically
# to make sure the db2nodes.cfg lines have the node numbers in ascending
# order).

package fvt_nodes_cfg;

@alphabet = ( 'A'..'Z' );
$STD_NAMES = 0;          # catalog is "c", rest are letters starting from A
$MLN_NAMES = 1;          #
$TRUE      = 1;
$FALSE     = 0;

sub init # ( $savename [, $mode | $fromsavename ] )
{
  local( $savename ) = shift @_;
  local( $mode, $fromsavename, @nodes, @nodenames, %networks,
@networks,
    $catalog, $name, $host, $port, $network, $newUsed, $",
  );

  #************************************************************
***************
  # Some things will do magic if INIT is set
  #************************************************************
***************
  &set( $savename, "INIT", 1 );

  $mode = $STD_NAMES; $fromsavename = undef;
  if( scalar( @_ ))
  {
    if( $_[0] =~ /^\d/ )
      { $mode = $_[0]; }
    else
      { $fromsavename = $_[0]; }
  }

  #************************************************************
***************
  # Only call this macro once per $savename...
  #************************************************************
***************
  unless( &get( $savename, "NODES" ) eq undef )
    { die "Only call fvt_nodes_cfg::init once for \"$savename\".\n"; }


  #************************************************************
***************
  # Name and process the lines of db2nodes.cfg.  If $fromsavename has been
  # set, use it as a base for the node names...
  #************************************************************
***************
  if( $fromsavename ne undef )
  {
    &set( $savename, "NAMING_CONVENTION", &get( $fromsavename,
"NAMING_CONVENTION" ));
    &set( $savename, "CATALOG_LINK", &get( $fromsavename,
"CATALOG_LINK" ));
    @processed = &name( $fromsavename, "N", "H", "P", "n" );
  }
  else
  {
    &set( $savename, "NAMING_CONVENTION", $mode );
    @processed = &name( $savename, "N", "H", "P", "n" );
  }

  while( scalar(@processed) )
  {
    $name    = shift( @processed );
    $network = shift( @processed );
    $host    = shift( @processed );
    $port    = shift( @processed );
    $netUsed = shift( @processed );

    push( @nodes, $name, "$host:$port:$netUsed" );
    push( @nodenames, "$host:$port", $name );
    if( $network )
      { $networks{$host} = $network; }
  }

  #************************************************************
***************
  # Store the results in environment variables, cleanup...
  #************************************************************
***************
  $" = " ";
  &set( $savename, "NODES", "@nodes" );
  &set( $savename, "NODE_NAMES", "@nodenames" );

  foreach $host (keys %networks)
    { push( @networks, $host, $networks{$host} ); }
  &set( $savename, "NETWORKS", "@networks" );

  &clear( $savename, "ORDERED" );

  #************************************************************
***************
  # The following two are "private" variables that allow &use to do insane and
  # illegal things.  If FREEDOM is set, using MLN_NAMES you can create
new
  # logical nodes on any available host.  If GARBAGE_NETWORK is set,
you can
  # supply the "g" flag with a node name to add a garbage network name.
  #************************************************************
***************
  &clear( $savename, "FREEDOM" );
  &clear( $savename, "GARBAGE_NETWORK" );

  #************************************************************
***************
  # Init the node numbers, and clear INIT
  #************************************************************
***************
  &nodenums( $savename );
  &clear( $savename, "INIT" );
}

# @attributes - NAMES, NUMBERS, HOSTS, PORTS
sub list # ( [$savename[, @namelist][, @attributes]] )
{
  local( $savename, @namelist ) = @_;
  local( $attributeName, @attributes, *data, $key, $name, @list, $i,
      @nameReqs, $offset );

  #************************************************************
***************
  # Grab the attributes  REMEMBER: You have to reverse the data before
returning it
  #************************************************************
***************
  if( scalar( @namelist ))
  {
    while( $namelist[$#namelist] =~ /^(NUMBERS|NAMES|HOSTS|PORTS|
NETWORKS)$/i)
    {
      pop( @namelist ); $attributeName = $1;

      $attributes[scalar( @attributes )] = (
        $attributeName eq "HOSTS"    ? "H" :
        $attributeName eq "PORTS"    ? "P" :
        $attributeName eq "NETWORKS" ? "N" :
        $attributeName eq "NAMES"    ? "_" :
          "#"
      );

      # Sorry.  Kludge for NAMES...
      $nameReqs[scalar(@nameReqs)] = $#attributes if $attributeName eq
"NAMES";
    }
  }
  unless( scalar( @attributes ) ) { $attributes[0] = "#"; }

  #************************************************************
***************
  # If $savename is undef, don't bother with names, just return the data...
  #************************************************************
***************
```

```
  if( $savename eq undef )
  {
    @attributes = reverse @attributes;
    return &slurp( @attributes );
  }


  #************************************************************** ***************
  # Load the relevant data
  #************************************************************** ***************
  @namelist = &resolve( $savename, @namelist );
  @data     = &name( $savename, @attributes );

  for( $i = 0; $i < scalar( @data ); $i += scalar( @attributes ) + 1 )
    { $data{$data[$i]} = $i+1; }

  # Because &name cannot put the NAME in the data list (without major
  # rework), handle it here.  Replace members of @data as noted by
@nameReqs
  foreach $offset (@nameReqs)
  {
    foreach $name (keys %data)
      { $data[$data{$name}+$offset] = $name; }
  }

  foreach $name (@namelist)
  {
    if( $data{$name} ne undef )
    {
      $offset = $data{$name};
      push( @list, reverse @data[$offset .. $offset + scalar( @attributes ) -
1 ] );
    }
  }

  return @list;
}

sub use # ( $savename, @namelist )
{
  local( $savename, @tmp_namelist, $" ) = @_;

  #************************************************************** ***************
  # Only call this macro after calling init for $savename
  #************************************************************** ***************
  if( &get( $savename, "NODES" ) eq undef )
    { die "Only call fvt_nodes_cfg::use for '$savename' after calling
fvt_nodes_cfg:init\n"; }

  #************************************************************** ***************
  # Setup the data structures...
  #************************************************************** ***************
  local( %refSet, %refSetNet, @namelist, @netstatelist, $item, @nodenums,
      @nodenumoverrides, *CFG, $i );

  foreach $item (@tmp_namelist)
  {
    ( $namelist[scalar(@namelist)], $netstatelist[scalar(@netstatelist)],
      $nodenumoverrides[scalar(@nodenumoverrides)] ) =
      split( /:/, $item, 3 );
  }

  @namelist  = &resolve( $savename, @namelist );
  %refSet    = split( / /, &get( $savename, "NODES" ));
  %refSetNet = split( / /, &get( $savename, "NETWORKS" ));

  @nodenums = &nodenums( $savename, @namelist );
  for( $i = 0; $i < scalar( @nodenumoverrides ); $i++ )
    { $nodenums[$i] = $nodenumoverrides[$i] if $nodenumoverrides[$i] ne
undef; }

  #************************************************************** ***************
  # Create the output lines...
  #************************************************************** ***************
  local( $name, $nodecfg, $netstate, @working, $netname, @lines );
  for( $i = 0; $i < scalar( @namelist ); $i++ )
  {
    ( $name, $netstate ) = ( $namelist[$i], $netstatelist[$i] );
    $nodecfg = $refSet{ $name };

    #************************************************************** *************
    # If invalid $name, and MLN_NODES and FREEDOM, try to create a new
node
    #************************************************************** *************
    if( $nodecfg eq undef
        && &get( $savename, "NAMING_CONVENTION" ) ==
$MLN_NAMES
        && &get( $savename, "FREEDOM" ) ne undef )
    {
      local( $host, $port, @working );
      if( ($host, $port) = $name =~ /(h[cA-Z][A-Z]*)(\d*)/ )
      {
        if( $port == 0 || $refSet{$host} eq undef )
          { die "fvt_nodes_cfg::use unable to create logical node '$name'\n"; }
        else
        {
          @working = split( /:/, $refSet{$host} );
          $nodecfg = "$working[0]:$port:$working[2]";
        }
      }
    }

    if( $nodecfg eq undef )
      { die "fvt_nodes_cfg::use unable to find node '$name'\n"; }

    @working = split( /:/, $nodecfg );


    #************************************************************** *************
    # Figure out the network name...
    # x - don't use it, n - use it, g - use garbage, if available, h - host
    #************************************************************** *************
    $netname =
      $netstate eq "x"                    ? "" :

      ($netstate eq "g" &&
       &get( $savename, "GARBAGE_NETWORK" ) ne undef) ? &get
( $savename, "GARBAGE_NETWORK" ) :

      $netstate eq "h"                    ? $working[0] :

      ($netstate eq "n" ||
       $working[2] == 1)                    ? $refSetNet{ $working[0] } :

                                        "";

    #************************************************************** *************
    # Finally, piece it all together...
    #************************************************************** *************
    @lines[scalar(@lines)] = sprintf( "%-6d %-15s %5d %s",
                          $nodenums[$i], @working[0..1], $netname );
  }

  #************************************************************** ***************
  # Write out the new config...
  #************************************************************** ***************
  @lines = sort numerically @lines unless( &get( $savename, "NO_SORT" ));
  $CFG = &main'get_db2nodes_cfg;
  chmod 0666, $CFG;
  open( CFG, ">$CFG" ) || die "fvt_nodes_cfg::use could not open
db2nodes.cfg\n";
  $" = "\n";
  print CFG "@lines\n";
  close( CFG );
}

# $qualifier:
#    -c   - remove the catalog from the list
```

```
#    <+l->#  - return the first (last) # nodes
sub resolve # ( $savename, @names [, $qualifier] )
{
  local( $savename, @names ) = @_;

  #************************************************************* ***************
  # Make sure there is something to resolve with...
  #************************************************************* ***************
  if( &get( $savename, "NODES" ) eq undef )
    { die "fvt_nodes_cfg::resolve: attempt to resolve from unknown savename
'$savename'\n"; }

  local( $mode, $qualifier );
  $mode = &get( $savename, "NAMING_CONVENTION" );

  #************************************************************* ***************
  # Grab the qualifier, if there is one...
  #************************************************************* ***************
  if( substr( $names[$#names], 1, 1 ) =~ /^[+-]/o )
    { $qualifier = pop( @names ); }

  #************************************************************* ***************
  # Process the list.
  #************************************************************* ***************
  if( scalar( @names ))
  {
    #************************************************************* **************
    # If the list is ("*"), expand "*", and recurse
    #************************************************************* **************
    if( $names[0] eq "*" )
    {
      local( @nodes );
      @nodes = split( / \S+\s?/, &get( $savename, "NODES" ));

      if( $qualifier )
        { @names = &resolve( $savename, @nodes, $qualifier ); }
      else
        { @names = &resolve( $savename, @nodes ); }
    }

    #************************************************************* **************
    # If the list is fully expanded, and there is no qualifier, convert the
    # "c" link (STD_NAMES is handled automatically...)
    #************************************************************* **************
    elsif( $qualifier eq undef )
    {
      local( $catalog, $name );
      $catalog = &get( $savename, "CATALOG_LINK" );

      if( $catalog )
      {
        foreach $name (@names)
        {
          if( $name eq "c" )
            { $name = $catalog; }
        }
      }
    }

    #************************************************************* **************
    # Otherwise, process the qualifier...
    #************************************************************* **************
    else
    {
      #************************************************************* ************
      # Recurse to resolve any "c" links
      #************************************************************* ************
      @names = &resolve( $savename, @names );

      #************************************************************* ************
      # If the qualifier is -c, delete all occurances of the catalog node
      #************************************************************* ***************
      if( $qualifier eq "-c" )
      {
        local( $catalog, %nodes );
        $catalog = &resolve( $savename, "c" );

        for( $i = 0; $i < scalar( @names ); $i++ )
        {
          if( $names[$i] eq $catalog )
            { @names = (@names[0..$i-1], @names[$i+1..$#names]); }
        }
      }
      #************************************************************* ************
      # Take a certain number of nodes from the front or back of the list
      #************************************************************* ************
      elsif( $qualifier =~ /^([+-]\d+)/o )
      {
        local( $value, $start, $end );

        $value = int $1;
        $start =  $value > 0 ? 0      : scalar( @list ) + $value;
        $end   = ($value > 0 ? $value : scalar( @list )        ) - 1;

        @names = @names[$start..$end];
      }
    }
  }

  return @names;
}

sub dump # ( $savename [, @nodelist] )
{
  local( $savename, @nodelist ) = @_;
  local( @lines );

  #************************************************************* ***************
  # If a nodelist was passed, create the dump from memory
  #************************************************************* ***************
  if( scalar( @nodelist ))
  {
    local( %refSet, %refSetNet, $node, $host, $port, $netUsed );

    @nodelist = &resolve( $savename, @nodelist );
    %refSet    = split( / /, &get( $savename, "NODES" ));
    %refSetNet = split( / /, &get( $savename, "NETWORKS" ));

    foreach $node (@nodelist)
    {
      ( $host, $port, $netUsed ) = split( /:/, $refSet{$node}, 3 );

      $lines[scalar(@lines)] = sprintf( "%-6s %-15s %5d %s",
        $node, $host, $port, $netUsed ? $refSetNet{$host} : "" );
    }
  }
  #************************************************************* ***************
  # If a nodelist was not passed, create the dump from disk
  #************************************************************* ***************
  else
  {
    local( @request, @processed, $i, $offset );
    @request = ($savename, "#", "H", "P", "N" );

    @processed = &name( @request );

    for( $i = 0, $offset = 0; $i < int(scalar(@processed) / scalar(@request));
        $i++, $offset = $i * scalar( @request ))
    {
      $lines[scalar(@lines)] = sprintf( "%-6s %-6d %-15s %5d %s",
        @processed[ $offset .. $offset + (scalar( @request )-1)] );
    }
```

```
  }

  return @lines;
}

sub set # ( $savename, $var, $value )
{
  local( $savename, $var ) = (shift @_, shift @_);
  local( $value ) = "@_";

  if( scalar( @_ ) <= 1 && @_[0] eq undef )
    { &clear( $savename, $var ); }
  else
    { $ENV{"FVT_NODES_CFG_${savename}_${var}"} = $value; }
}

sub get # ( $savename, $var )
{
  local( $savename, $var ) = @_;

  return $ENV{"FVT_NODES_CFG_${savename}_${var}"};
}

sub clear # ( $savename, $var )
{
  local( $savename, $var ) = @_;

  unless( $var eq undef )
    { delete $ENV{"FVT_NODES_CFG_${savename}_${var}"}; }
  else
  {
    foreach $key (keys %ENV)
    {
      if( $key =~ /^FVT_NODES_CFG_${savename}_/ )
        { delete $ENV{$key}; }
    }
  }
}


sub nodenums # ( $savename, @names )
{
  local( $savename, @names ) = @_;
  local( $order, $name );

  #*************************************************************
***************
  # Figure out which number set to use...
  #*************************************************************
***************
  if( &get( $savename, "INIT" ) ne undef )
    { $name = "RANDOM"; }
  else
  {
    if( &get( $savename, "FREEDOM" ) ne undef )
      { $name = "FREEDOM "; }

    if( ($order = &get( $savename, "ORDERED" )) ne undef )
      { $name .= "ORDERED $order"; }
    else
      { $name .= "RANDOM"; }
  }

  #*************************************************************
***************
  # Psuedo-kludge to allow multiple sets of node numbers
  #*************************************************************
***************
  if( &get( $savename, "NEW_NODE_NUMBERS" ) ne undef )
  {
    local( $data ) = &get( $savename, "NUMBERING" );

    if( $data =~ /$name/ )
    {
      $data =~ s/$name\n((h?[cA-Z][A-Z]*[0-9]* \d+ )*)\n//;
      &set( $savename, "NUMBERING", $data );
    }
  }

  #*************************************************************
***************
  # If necessary, setup the number set...
  #*************************************************************
***************
  if( &get( $savename, "NUMBERING" ) !~ /$name/ )
  {
    local( @data, @nodenums, @nodenames, $i, $data );

    if( $name eq "RANDOM" )
    {
      # This is the first use, and the numbers are in the file...
      @data = &list( $savename, "*", "NUMBERS", "NAMES" );

      for( $i = 0; $i < scalar( @data ); $i += 2 )
      {
        $nodenums[scalar(@nodenums)] = $data[$i];
        $nodenames[scalar(@nodenames)] = $data[$i+1];
      }
    }
    else
    {
      @nodenames = @names;
      @nodenums  = &makenodenums( $savename, scalar( @nodenames ));
    }

    for( $i = 0, $data = ""; $i < scalar( @nodenames ); $i++ )
      { $data .= "$nodenames[$i] $nodenums[$i] "; }

    &set( $savename, "NUMBERING",
        ( &get( $savename, "NUMBERING" ) . "$name\n$data\n" ));
  }

  return () if( &get( $savename, "INIT" ) ne undef );

  #*************************************************************
**************
  # Now, grab the complete number set, and return the requested ones...
  #*************************************************************
**************
  local( *data, $node, @nodenums );

  $data = &get( $savename, "NUMBERING" );
  $data =~ /$name\n((h?[cA-Z][A-Z]*[0-9]* \d+ )*)\n/;
  $data = $1;

  %data = split( / /, $data );

  foreach $node (@names)
  {
    $nodenums[scalar(@nodenums)] =
      $data{$node} ? $data{$node}
                : "unknown node: ${savename}::${node}";
  }

  return @nodenums;
}

sub makenodenums # ( $savename, $count )
{
  local( $savename, $count ) = @_;
  local( @nodenums, @random, $i, $range, $time, @ordered );

  #*************************************************************
***************
  # Figure out what we're doing.  If we're doing nothing, return ()
  #*************************************************************
***************
  if( $count == 0 )
    { return (); }

  @ordered = split( / /, &get( $savename, "ORDERED" ) );

  #*************************************************************
***************
  # If ordering is random, create list of $count random node numbers...
  #*************************************************************
***************
  if( scalar(@ordered) == 0 )
  {
    $time = time;
    srand ($time & 0xffff);
    $range = int( 999 / $count );

    for( $i = 0; $i < $count; $i++ )
```

```
      { $random[scalar(@random)] = rand( $range ) * 100 % $range; }

    #*************************************************************
**************
    # Include two types of random -- maximum random and maximum
coverage...
    #*************************************************************
**************
    for( $i = 0; $i < $count; $i++ )
    {
      if( $time % 2 )
        { $nodenums[$i] = $random[$i] + int( $i * $range ); }
      else
      {
        $nodenums[$i] = $nodenums[$i-1] + $random[$i];
        $nodenums[$i]++ if( $i && $nodenums[$i] == $nodenums[$i-1] );
      }
    }
  }
  #*************************************************************
***************
  # ... else, create a set of ordered numbers, based on the start and step in
  # @ordered...
  #*************************************************************
***************
  else
  {
    for( $i = 0; $i < $count; $i++ )
      { $nodenums[$i] = $ordered[0] + ($i * $ordered[1]); }
  }

  return @nodenums;
}

sub slurp # ( @fields )
{
  local( @fields ) = @_;
  local( *CFG, @lines, $line, $field, $num, $host, $port, $network, @details );

  #*************************************************************
***************
  # Slurp in db2nodes.cfg...
  #*************************************************************
***************
  open( CFG, &main'get_db2nodes_cfg ) || die "fvt_nodes_cfg::slurp could not
open db2nodes.cfg\n";
  chop( @lines = <CFG> );
  close( CFG );

  #*************************************************************
***************
  # Sort the nodes file by host and port (to ensure the naming system
  # produces intuitive results)
  #*************************************************************
***************
  @lines = &sortcfg( @lines );

  #*************************************************************
***************
  # Process each line in turn...
  #*************************************************************
***************
  @details = ();
  foreach $line (@lines)
  {
    ( $num, $host, $port, $network ) = split( /[ \t]+/, $line, 4 );
    foreach $field (@fields)
    {
      push( @details,
        $field eq "#" ? $num          :
        $field eq "H" ? $host         :
        $field eq "P" ? $port         :
        $field eq "N" ? $network      :
        $field eq "n" ? ($network?1:0) :
        $field eq "_" ? ""            :
                "fvt_nodes_cfg::process UNKNOWN FIELD: $field"
      );
    }
  }

  return @details;
}

# Either matches to a current set, or makes a new set...
sub name # ( $savename, @fields )
{
  local( $savename, @fields ) = @_;
  local( $new, @anchorFields, @parsed, $fieldsPerRow, $actualPerRow, $i,
      @fieldValues, @anchorValues, @list, $nodename, $mode, $host, $port,
  );

  $new = $FALSE;
  $new = $TRUE if &get( $savename, "NODES" ) eq undef;

  #*************************************************************
***************
  # Process the db2nodes.cfg file...
  #*************************************************************
***************
  @anchorFields = ( "H", "P" );
  unshift( @anchorFields, "#" ) if $new;

  @parsed = &slurp( @fields, @anchorFields );

  $fieldsPerRow = scalar( @fields );
  $actualPerRow = $fieldsPerRow + scalar( @anchorFields );

  #*************************************************************
***************
  # Set up for each type of processing...
  #*************************************************************
***************
  $mode = &get( $savename, "NAMING_CONVENTION" );

  local( %refSet, ) if !$new;
  if( !$new )
  {
    %refSet = split( / /, &get( $savename, "NODE_NAMES" ));
  }

  local( $catalogNum, $catalogHost, $nameMajor, $nameMinor, $num, %
hosts,
      %hostCount, $catalogLink, ) if $new;
  if( $new )
  {
    $catalogNum = &main'fvt_get_catalog_node;
    $nameMajor = -1;
    $nameMinor = 0;

    #*************************************************************
**************
    # If we are using MLN_NAMES, we need to know the catalog host first...
    #*************************************************************
**************
    $catalogLink = "c";
    if( $mode == $MLN_NAMES )
    {
      $catalogLink = undef;
      for( $i = 0, $offset = 0; $i < int( scalar(@parsed) / $actualPerRow );
$i++, $offset = $i * $actualPerRow )
      {
        ( $num, $host, $port ) = @parsed[ ($offset + $fieldsPerRow) ..
                                  ($offset + $actualPerRow - 1)];

        if( $num == $catalogNum )
        {
          $catalogHost = $host;
          last;
        }
      }
    }
  }


  #*************************************************************
***************
  # Name each set of data, putting the result on @list
  #*************************************************************
***************
  for( $i = 0, $offset = 0; $i < int( scalar(@parsed) / $actualPerRow );
      $i++, $offset = $i * $actualPerRow )
  {
    @fieldValues  = @parsed[ $offset .. ($offset + $fieldsPerRow - 1) ];
```

```
    @anchorValues = @parsed[ ($offset + $fieldsPerRow) ..
                     ($offset + $actualPerRow - 1)];

    #*********************************************************
**************
    # Assign names based on existing configuration...
    #*********************************************************
**************
    if( !$new )
    {
      ( $host, $port ) = @anchorValues;
      $nodename = $refSet{ "$host:$port" };

      #*******************************************************
************
      # If the nodename was not found, and FREEDOM is set, try to create
      # the name
      #*******************************************************
************
      if( $nodename eq undef
          && $mode == $MLN_NAMES && &get( $savename, "FREEDOM"
) )
      {
        $nodename = $refSet{ "$host:0" };
        $nodename .= $port if $nodename ne undef;
      }

      #*******************************************************
************
      # If no nodename could be found, die.  Otherwise, save the data...
      #*******************************************************
************
      if( $nodename eq undef )
        { die "fvt_nodes_cfg::name unknown node name for node
$host:$port\n"; }

      push( @list, $nodename, @fieldValues );
    }

    #*********************************************************
**************
    # Assign new names
    #*********************************************************
**************
    else
    {
      ( $num, $host, $port ) = @anchorValues;

      #*******************************************************
************
      # Handle STD_NAMES ...
      #*******************************************************
************
      if( $mode == $STD_NAMES )
      {
        $nodename = $num == $catalogNum
                  ? "c"
                  : "$alphabet[$nameMajor]$alphabet[$nameMinor]";
        push( @list, $nodename, @fieldValues );

        if( $num != $catalogNum )
        {
          $nameMinor = ($nameMinor+1) % 26;
          $nameMajor++ if $nameMinor == 0;
        }
      }
      #*******************************************************
************
      # ... and MLN_NAMES separately...
      #*******************************************************
************
      else
      {
        #*****************************************************
***********
        # Create a name for the host...
        #*****************************************************
***********
        unless( $hosts{$host} )
        {
          $hosts{$host} =
            $host eq $catalogHost
                  ? "hc"
                  : "h$alphabet[$nameMajor]$alphabet[$nameMinor]";

          if( $host ne $catalogHost )
          {
            $nameMinor = ($nameMinor+1) % 26;
            $nameMajor++ if $nameMinor == 0;
          }
        }

        #*****************************************************
**********
        # Name the node...
        #*****************************************************
**********
        $nodename = $port == 0
                  ? $hosts{$host}
                  : $hosts{$host} . (++$hostCount{$host});
        push( @list, $nodename, @fieldValues );

        #*****************************************************
**********
        # If its the catalog node, add a "symbolic link"
        #*****************************************************
**********
        if( $num == $catalogNum )
          { $catalogLink = $nodename; }
      }

      &set( $savename, "CATALOG_LINK", $catalogLink );
    }
  }

  return @list;
}

sub sortcfg # ( @lines )
{
  local( @lines ) = @_;
  local( %anchor, %compare, $trash );

  #*********************************************************
***************
  # Move through the lines, comparing each line to all that follow.  Swap any
  # lines where the host name matches, but the earlier line has a larger port
  # number than the later line.
  #*********************************************************
***************
  $anchor{'index'} = -1;
  while( ++$anchor{'index'} < scalar( @lines ))
  {
    $anchor{'line'} = $lines[$anchor{'index'}];
    ( $trash, $anchor{'host'}, $anchor{'port'}, $trash ) = split( /[ \t]+/, $anchor
{'line'}, 4 );

    #*******************************************************
*************
    # A simple optimization.  If the port is 0, it will not be moving...
    #*******************************************************
*************
    next if $anchor{'port'} == 0;

    $compare{'index'} = $anchor{'index'};
    while( ++$compare{'index'} < scalar( @lines ))
    {
      $compare{'line'} = $lines[$compare{'index'}];
      ( $trash, $compare{'host'}, $compare{'port'}, $trash ) = split( /[ \t]+/,
$compare{'line'}, 4 );

      if( $anchor{'host'} eq $compare{'host'} )
      {
        if( $anchor{'port'} > $compare{'port'} )
        {
          #*****************************************************
********
          # Swap the compared lines in @lines.  Then update %anchor, and
          # continue as if nothing had happened...
          #*****************************************************
********
          $lines[$anchor{'index'}]  = $compare{'line'};
          $lines[$compare{'index'}] = $anchor{'line'};
          %anchor = %test;
```

```
        }
      }
    }
  }

  return @lines;
}

sub numerically
  { $a <=> $b; }

package main;


##External
#Package fvt_nodes_cfg_macros
#
# This package contains various functions to make common operations with
the
# fvt_nodes_cfg package easier to do.
#
# ordernodes()
#   Reorder the node numbers in the current node configuration to start at 0
#   and increase by 1.
#
package fvt_nodes_cfg_macros;
$TempSave = "FVCMACROTEMP";

sub ordernodes
{
  &fvt_nodes_cfg'init( $TempSave, $fvt_nodes_cfg'MLN_NAMES );
  &fvt_nodes_cfg'set( $TempSave, "ORDERED", "0 1" );
  &fvt_nodes_cfg'use( $TempSave, "*" );
  &fvt_nodes_cfg'clear( $TempSave );
}

package main;

1;
#
```

tpcdmacro.pl

```
#!/usr/bin/perl
# subroutine to execute and check return codes for db2 commands that require
a
# connection
sub dodb_conn
{
 local($dbname,$cmd,$all_nodes) = @_;
 local($ret) = 0;

 if ( $all_nodes eq "all_ln" )
 {
  if ( $platform eq "nt" ) {
   $ret=system("db2_all \"|| db2 connect to ${dbname} & ${cmd} & db2
connect reset & db2 terminate \" ");
  }
  elsif ($platform eq "sun") {
     $ret=system("db2_all \'; db2 connect to ${dbname}; ${cmd}; db2
connect reset; db2 terminate \' ");
  }
  elsif ($platform eq "linux") {
            $ret=system("db2_all \'db2 connect to ${dbname}; ${cmd}; db2
connect reset; db2 terminate \' ");
  }
  else {
     $ret=system("db2_all \'||}] db2 connect to ${dbname}; ${cmd}; db2
connect reset; db2 terminate \' ");
  }
 }
 else
 {
  if ( $platform eq "nt" ) {
   open(FILE, ">dodb_conn.tmp.bat");
   print FILE "db2 connect to ${dbname} & ${cmd} & db2 connect reset &
db2 terminate";
   close(FILE);
   $ret=system("dodb_conn.tmp");
   system("del dodb_conn.tmp.bat");
  }
  else {
   $ret=system("db2 connect to ${dbname}; ${cmd}; db2 connect reset; db2
terminate ");
  }
 }
 return($ret);
}

# subroutine to execute and check return codes for db2 commands that do not
# require a connection
sub dodb_noconn
{
 local($cmd,$all_nodes) = @_;
 local($ret) = 0;

 if ( $all_nodes eq "all_ln" )
 {
  if ( $platform eq "nt" ) {
   $ret=system("db2_all \"|| ${cmd} \" ");
  }
  elsif ( $platform eq "sun" ) {
     $ret=system("db2_all \'; ${cmd} \' ");
  }
  elsif ($platform eq "linux") {
            $ret=system("db2_all \'${cmd} \' ");
  }
  else {
   $ret=system("db2_all \'||}] ${cmd} \' ");
  }
 }
 else
 {
  if ( $platform eq "nt" ) {
   open(FILE, ">dodb_noconn.tmp.bat");
   print FILE "${cmd}";
   close(FILE);
   $ret=system("dodb_noconn.tmp");
   system("del dodb_noconn.tmp.bat");
  }
  else {
   $ret=system(" ${cmd} ");
  }
 }
 return($ret);
}

# execute a file in db2
sub dodb2file
{
 local($dbname,$fname,$all_nodes) = @_;
 local($ret) = 0;

 if ( $all_nodes eq "all_ln" )
 {
  if ( $platform eq "nt" ) {
   $ret=system("db2_all \"|| db2 connect to ${dbname} & db2 $db2options -f
$fname & db2 connect reset & db2 terminate \" ");
  }
  elsif ( $platform eq "sun" ) {
     $ret=system("db2_all \'; db2 connect to ${dbname}; db2 $db2options -f
$fname; db2 connect reset; db2 terminate \' ");
  }
  elsif ($platform eq "linux") {
            $ret=system("db2_all \'db2 connect to ${dbname}; db2
$db2options -f $fname; db2 connect reset; db2 terminate \' ");
  }
  else {
   $ret=system("db2_all \'||}] db2 connect to ${dbname}; db2 $db2options -f
$fname; db2 connect reset; db2 terminate \' ");
  }
 }
 else
 {
  if ( $platform eq "nt" ) {
   open(FILE, ">dodb2file.tmp.bat");
   print FILE "db2 connect to ${dbname} & db2 $db2options -f $fname &
db2 connect reset & db2 terminate";
   close(FILE);
   $ret=system("dodb2file.tmp");
   system("del dodb2file.tmp.bat");
```

```
    }
    else {
      $ret=system("db2 connect to ${dbname}; db2 $db2options -f $fname; db2
connect reset; db2 terminate ");
    }
  }
  return($ret);
}

# execute a file in db2 that doesn't need a connection
sub dodb2file_noconn
{
  local($fname,$all_nodes) = @_;
  local($ret) = 0;

  if ( $all_nodes eq "all_ln" )
  {
    if ( $platform eq "nt" ) {
      $ret=system("db2_all \"|| db2 $db2options -f $fname & db2 terminate \" ");
    }
    elsif ( $platform eq "sun" ) {
        $ret=system("db2_all \'; db2 $db2options -f $fname; db2 terminate \' ");
    }
    elsif ( $platform eq "linux") {
        $ret=system("db2_all \db2 $db2options -f $fname; db2 terminate \' ");
    }
    else {
      $ret=system("db2_all \||}] db2 $db2options -f $fname; db2 terminate \' ");
    }
  }
  else
  {
    if ( $platform eq "nt" ) {
      open(FILE, ">dodb2file.tmp.bat");
      print FILE "db2 $db2options -f $fname & db2 terminate";
      close(FILE);
      $ret=system("dodb2file.tmp");
      system("del dodb2file.tmp.bat");
    }
    else {
      $ret=system("db2 $db2options -f $fname; db2 terminate ");
    }
  }
  return($ret);
}

sub outtime
{
    local($stmt) = @_;
    local($current_time) = 0;

    $current_time = `perl gettimestamp long`;
    print "$stmt at : $current_time\n ";
}

sub stopStart{

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

sub flush {
    local($old) = select(shift);
    $| = 1;
    print "";
    $| = 0;
    select($old);
}

sub printflush {
    local($old) = select(shift);
    $| = 1;
    print @_;
    $| = 0;
    select($old);
}

1;
```

buildtpcdbatch

```
#!/usr/bin/perl
# usage  buildtpcdbatch [QUAL]
($myName = $0) =~ s@.*/@@; $usage="
Usage: buildtpcd [QUAL]
    where QUAL is the optional parameter saying to bind to the qualification
            database (sf = .1 = 100MB)\n";

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


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

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

#verify environment variables required for script to run
@reqVars     = ("TPCD_PRODUCT",
                "TPCD_MODE",
                "TPCD_AUDIT_DIR",
                "TPCD_DBNAME",
                "TPCD_PATH_DELIM",
                "TPCD_PLATFORM",
                "TPCD_HAVECOMPILER");

&setVar(@reqVars, "ERROR");

#set up local variables
$auditDir=          $ENV{"TPCD_AUDIT_DIR"};
$dbname= $ENV{"TPCD_DBNAME"};
$delim=             $ENV{"TPCD_PATH_DELIM"};
$platform=          $ENV{"TPCD_PLATFORM"};
$havecompiler=      $ENV{"TPCD_HAVECOMPILER"};
$product= $ENV{"TPCD_PRODUCT"};
$mode=              $ENV{"TPCD_MODE"};

if(($platform eq "ptx") || ($platform eq "linux") || ($platform eq "hp"))
{
    $Tproduct=$platform;
}
else
{
    $Tproduct="unix";
}

# set up override of some parameters to override for qualification database
if ( $qual eq "QUAL" )
{
    if ( $ENV{"TPCD_QUAL_DBNAME"} eq "NULL" )
    {
        die "TPCD_QUAL_DBNAME environment variable not set\n";
    }
    $qualdbname=$ENV{"TPCD_QUAL_DBNAME"};
}

# links the implementation specific layer code into the auditruns directory
#cd $auditDir/auditruns
if ( $platform eq "nt" )
{
    if ( $havecompiler eq "yes" )
    {
        &cp("$auditDir${delim}driver${delim}tpcdbatch.sqc", "$auditDir$
{delim}auditruns${delim}tpcdbatch.sqc");
        &cp("$auditDir${delim}driver${delim}tpcdbatch.h", "$auditDir${delim}
```

```
auditruns${delim}tpcdbatch.h");
      &cp("$auditDir${delim}driver${delim}tpcdUF.sqc", "$auditDir${delim}
auditruns${delim}tpcdUF.sqc");
      &cp("$auditDir${delim}driver${delim}tpcd_bnd.bat", "$auditDir$
{delim}auditruns${delim}tpcd_bnd.bat");
      &cp("$auditDir${delim}driver${delim}tpcd_cl.bat", "$auditDir${delim}
auditruns${delim}tpcd_cl.bat");
      &cp("$auditDir${delim}driver${delim}tpcdqual_bnd.bat", "$auditDir$
{delim}auditruns${delim}tpcdqual_bnd.bat");
      &cp("$auditDir${delim}driver${delim}tpcdqual_cl.bat", "$auditDir$
{delim}auditruns${delim}tpcdqual_cl.bat");
      chdir("$auditDir${delim}auditruns");
      if ( $qual ne "QUAL" )
      {
         system("tpcd_cl $dbname");
         system("tpcd_bnd $dbname");
      }
      else
      {
         system("tpcdqual_cl $qualdbname");
         system("tpcdqual_bnd $qualdbname");
      }
   }
   else
   {
      &cp("$auditDir${delim}driver${delim}tpcdbatch.sqc", "$auditDir$
{delim}auditruns${delim}tpcdbatch.sqc");
      &cp("$auditDir${delim}driver${delim}tpcdbatch.h", "$auditDir${delim}
auditruns${delim}tpcdbatch.h");
      &cp("$auditDir${delim}driver${delim}tpcdUF.sqc", "$auditDir${delim}
auditruns${delim}tpcdUF.sqc");
      &cp("$auditDir${delim}driver${delim}tpcdbatch.exe", "$auditDir$
{delim}auditruns${delim}tpcdbatch.exe");
      &cp("$auditDir${delim}driver${delim}tpcdbatch.bnd.nt", "$auditDir$
{delim}auditruns${delim}tpcdbatch.bnd");
      &cp("$auditDir${delim}driver${delim}tpcd_bnd.bat", "$auditDir$
{delim}auditruns${delim}tpcd_bnd.bat");
      &cp("$auditDir${delim}driver${delim}tpcdqual_bnd.bat", "$auditDir$
{delim}auditruns${delim}tpcdqual_bnd.bat");
      chdir("$auditDir${delim}auditruns");
      if ( $qual eq "QUAL" )
      {
         system("tpcdqual_bnd.bat $qualdbname");
      }
      else
      {
         system("tpcd_bnd.bat $dbname");
      }
   }
}
elsif ( $platform eq "os2" )
{
  die "[$0] Make tpcdbatch not defined for OS/2 yet!\n"
}
else     #this catches aix, other unix platforms and everything else!
{
   system("ln -sf $auditDir/driver/tpcdbatch.sqc
$auditDir/auditruns/tpcdbatch.sqc");
   system("ln -sf $auditDir/driver/tpcdbatch.sqc
$auditDir/auditruns/tpcdbatch.h");
   system("ln -sf $auditDir/driver/tpcdbatch.sqc
$auditDir/auditruns/tpcdUF.sqc");
   system("ln -sf $auditDir/driver/Makefile.$Tproduct
$auditDir/auditruns/Makefile.$Tproduct");
   if ( $havecompiler eq "yes" )
   {
    # compiles tpcdbatch  (assumes you have the database manager started,
    # but no database connection)
    system("cd $auditDir/driver;make -f Makefile.$Tproduct cleanup");
    system("cd $auditDir/driver;make -f Makefile.$Tproduct tpcdbatch");
    system("ln -sf $auditDir/driver/tpcdbatch $auditDir/auditruns/tpcdbatch");
   }
   else
   {
      if (( $mode eq "uni" ) || ( $mode eq "smp" ))
      {
         system("ln -sf $auditDir/driver/tpcdbatch
$auditDir/auditruns/tpcdbatch");
         system("cd $auditDir/driver;db2 connect to $dbname;db2 bind
tpcdbatch.bnd ISOLATION RR BLOCKING ALL;db2 connect reset; db2
terminate");
      }
      else
      {
         # mln and mpp need to use a differently compiled version
         system("ln -sf $auditDir/driver/tpcdbatch.mln
$auditDir/auditruns/tpcdbatch");
         system("cd $auditDir/driver;db2 connect to $dbname;db2 bind
tpcdbatch.mln.bnd ISOLATION RR BLOCKING ALL;db2 connect reset; db2
terminate");
      }
   }
   if ( $qual eq "QUAL" )
   {
    if (( $mode eq "uni" ) || ( $mode eq "smp" ) || ( $havecompiler eq "yes" ))
    {
       # bind tpcd batch to the qualification db also...use vanilla bind file
       # if this is uni or smp, or if we have done the compilation of tpcdbatch
       # on the current machine
      system("cd $auditDir/driver;db2 connect to $qualdbname;db2 bind
tpcdbatch.bnd ISOLATION RR BLOCKING ALL;db2 connect reset; db2
terminate");
    }
    else
    {
       # bind tpcd batch to the qualification db also...use tpcdbatch.mln.bnd
       # if we haven't got a compiler
      system("cd $auditDir/driver;db2 connect to $qualdbname;db2 bind
tpcdbatch.mln.bnd ISOLATION RR BLOCKING ALL;db2 connect reset; db2
terminate");
    }
   }
}

exit;
#
```

Appendix E: ACID Transaction Source Code

Makefile

```
DBNAME=         $(TPCD_QUAL_DBNAME)

INCLUDE =       $(HOME)/sqllib/include

CFLAGS  =       -I$(INCLUDE) -g -Dpascal= -DLINT_ARGS  \
        -DSQLA_NOLINES  -DLinux

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
```

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;
  int     o_key;
  double  l_extendedprice;
};

struct acidT_struct {
  int     termination;
  int     tag;
  int     logging;
  int     o_key;
  int     l_key;
  int     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[10];   /* = "tpcd"; */
EXEC SQL END DECLARE SECTION;

#ifdef SQLWINT

 /*
 ** redefine gettimeofday so I don't have to
 ** change too much aix-specific code
 */
/*#typedef struct timeval { unsigned tv_sec; unsigned tv_usec; }; */
typedef struct timezone { int dummy; };
struct timeb timer;

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

/*-----------------------------------------------------------*/
/*        acidQ                                              */
/*-----------------------------------------------------------*/
int acidQ (struct acidQ_struct *acid)
{
  time_t timeT;
  FILE *out;
  char out_fn[256];
  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_EXTENDEDPRICE,20,3)), 20,3) *
        (1-L_DISCOUNT)) * (1+L_TAX)),20,3)/100.0),20,3) * 1000)
      into :lEprice
    FROM
      TPCD.LINEITEM
    WHERE
      L_ORDERKEY = :okey;

  if (sqlca.sqlcode != 0) {
    rc = sqlca.sqlcode;
    fprintf(out,"acidQ **ERROR** sqlcode = %d\n",sqlca.sqlcode);
    sqlerror("acidQ: select sum(l_extendedprice)", &sqlca);
    goto Qerror;
  }
  eprice = (double)lEprice / 1000.0;  /* translate to double for printout*/

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

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

  rc = 0;
  goto Qexit;

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

 Qexit:
  fprintf(out,"\n---------- END of acidQ tag: %d ----------\n\n",mypid);
  fflush(out);fclose(out);
  return(rc);
}

/*-----------------------------------------------------------*/
/*        ast_acidQ                                          */
/*-----------------------------------------------------------*/
int ast_acidQ (struct acidQ_struct *acid)
{
  time_t timeT;
  FILE *out;
  char out_fn[256];
  struct timeval tv;
  struct timezone tz;
  int mypid;
  int rc = 0;


  EXEC SQL BEGIN DECLARE SECTION;
  double    ast_lEprice;
  double    ast_eprice;
  EXEC SQL END DECLARE SECTION;
```

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

  sprintf(out_fn, "%s%cast_acidQ.out.%d",getenv("TPCD_TMP_DIR"),del(),
mypid);
  out=fopen(out_fn,"a");
  gettimeofday(&tv, &tz);
  time(&timeT);
  fprintf(out,"\n---------- START of ast_acidQ tag: %d ----------\n\n",mypid);
  fprintf(out, "ast_acidQ tag: %d, begin transaction time: (%us %06uu) %s",
        mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));

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

  /*
  ** use the same query acidQ except don't select for specfic okey.
  ** this ensures that the ast will be used instead of the base table
  ** Have to use ast_lEprice as double since this sum is so big
  */
  EXEC SQL
    SELECT
      SUM ( L_EXTENDEDPRICE*(1-L_DISCOUNT)*(1 + L_TAX))
      into :ast_lEprice
    FROM
      TPCD.LINEITEM;

  if (sqlca.sqlcode != 0) {
    rc = sqlca.sqlcode;
    fprintf(out,"ast_acidQ **ERROR** sqlcode = %d\n",sqlca.sqlcode);
    sqlerror("ast_acidQ: select sum(l_extendedprice)", &sqlca);
    goto Qerror;
  }
  ast_eprice = ast_lEprice;  /* use ast_eprice  for printout to be consistent*/

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

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

  rc = 0;
  goto Qexit;

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

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

  time_t timeT;
  FILE *out;
  char out_fn[256];
  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, ototal;
  EXEC SQL END DECLARE SECTION;

  EXEC SQL DECLARE l_cursor CURSOR FOR
    SELECT l_partkey, l_suppkey, l_quantity,
    l_tax, l_discount,
    l_extendedprice
      FROM tpcd.lineitem
      WHERE l_orderkey = :o_key
      AND l_linenumber = :l_key
      FOR UPDATE OF l_extendedprice, l_quantity;

  EXEC SQL DECLARE o_cursor CURSOR FOR
    SELECT o_totalprice
      FROM tpcd.orders
      WHERE o_orderkey = :o_key
      FOR UPDATE OF o_totalprice;

  if (acid->termination < 0 || acid->termination > 3) acid->termination = 0;
  o_key = acid->o_key;
  l_key = acid->l_key;
  delta = acid->delta;

  if (acid->logging) {
    /* mypid = getpid(); */
    mypid = acid->tag;
    sprintf(out_fn, "%s%cacidT.out.%d",getenv("TPCD_TMP_DIR"),del(),
mypid);
    out=fopen(out_fn,"a");
    gettimeofday(&tv, &tz);
    time(&timeT);
    fprintf(out,"\n---------- START of acidT tag: %d ----------\n\n",mypid);
    fprintf(out, "acidT tag: %d, begin transaction time: (%us %06uu) %s",
          mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    fprintf(out, "o_key: %d\tl_key: %d\tdelta: %d\n", o_key, l_key, delta);
  }
#ifdef DEBUG
  printf("o_key: %d\tl_key: %d\tdelta: %d\n", o_key, l_key, delta);
#endif

 retry_tran:

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

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

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

#ifdef DEBUG
```

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

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

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

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

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

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

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

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

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

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

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

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

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

#ifdef DEBUG
  {
   double zeroone= l_extendedprice * (1.0- l_discount);
   double zeroonetimes= (l_extendedprice * (1.0- l_discount))*100.0;
   double firstone = TRUNC2(l_extendedprice * (1.0-l_discount));
   double notone= TRUNC2 ( l_extendedprice * (1.0-l_discount)) *
(1.0+l_tax);
   double secondone= TRUNC2( TRUNC2( l_extendedprice * (1.0-
l_discount) ) * (1.0+l_tax) );
   printf("firstone=   %f\n", firstone);
   printf("zeroone=   %f\n", zeroone);
   printf("zeroonetimes=   %f\n", zeroonetimes);
   printf("notone=   %f\n", notone);
   printf("secondone=   %f\n", secondone);
  }
#endif
  ototal = o_totalprice -
            TRUNC2( TRUNC2( l_extendedprice * (1-l_discount) ) *
(1+l_tax) );
  new_ototal = TRUNC2( new_extprice * (1.0-l_discount) );
  new_ototal = TRUNC2( new_ototal * (1.0+l_tax) );
  new_ototal = ototal + new_ototal;

#ifdef DEBUG
  printf("o_totalprince= %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:
    /* gjc@011214 */
    if (acid->logging) {
      gettimeofday(&tv, &tz);
      time(&timeT);
      fprintf(out,"acidT tag: %d, immediately before closing cursors: (%us %
06uu) %s",
            mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    }
    EXEC SQL CLOSE L_CURSOR;
    /* gjc@011214. */
    if (sqlca.sqlcode != 0) {
      if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
      rc = sqlca.sqlcode;
      if (acid->logging) {
        fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } else {
        fprintf(stderr,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } /* endif */
      sqlerror("acidT: Close L_CURSOR", &sqlca);
      goto Terror;
    }

    EXEC SQL CLOSE O_CURSOR;
    /* gjc@011214. */
    if (sqlca.sqlcode != 0) {
      if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
      rc = sqlca.sqlcode;
      if (acid->logging) {
        fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } else {
        fprintf(stderr,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } /* endif */
      sqlerror("acidT: Close O_CURSOR", &sqlca);
      goto Terror;
    }

    if (acid->logging) {
      gettimeofday(&tv, &tz);
      time(&timeT);
      fprintf(out,"acidT tag: %d, immediately before COMMIT: (%us %06uu)
%s",
            mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    }
    EXEC SQL COMMIT;
    if (sqlca.sqlcode != 0) {
      if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
      rc = sqlca.sqlcode;
      if (acid->logging) {
        fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } else {
        fprintf(stderr,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } /* endif */
      sqlerror("acidT: COMMIT", &sqlca);
      goto Terror;
    }
    if (acid->logging) {
      gettimeofday(&tv, &tz);
      time(&timeT);
      fprintf(out,"acidT tag: %d, after COMMIT: (%us %06uu) %s",
            mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    }
    break;
  case 3:
    if (acid->logging) {
      gettimeofday(&tv, &tz);
      time(&timeT);
      fprintf(out,"acidT tag: %d, wait before ROLLBACK: (%us %06uu) %s",
            mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    }
    sleep(60);
  case 2:
    if (acid->logging) {
      gettimeofday(&tv, &tz);
      time(&timeT);
      fprintf(out,"acidT tag: %d, immediately before closing cursors: (%us %
06uu) %s",
            mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    }
    EXEC SQL CLOSE L_CURSOR;
    /* gjc@011214. */
    if (sqlca.sqlcode != 0) {
      if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
      rc = sqlca.sqlcode;
      if (acid->logging) {
        fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } else {
```

```
        fprintf(stderr,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } /* endif */
      sqlerror("acidT: Close L_CURSOR", &sqlca);
      goto Terror;
    }
    EXEC SQL CLOSE O_CURSOR;
    /* gjc@011214. */
    if (sqlca.sqlcode != 0) {
      if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
      rc = sqlca.sqlcode;
      if (acid->logging) {
        fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } else {
        fprintf(stderr,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } /* endif */
      sqlerror("acidT: Close O_CURSOR", &sqlca);
      goto Terror;
    }

    if (acid->logging) {
      gettimeofday(&tv, &tz);
      time(&timeT);
      fprintf(out,"acidT tag: %d, immediately before ROLLBACK: (%us %
06uu) %s",
            mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    }
    EXEC SQL rollback work;
    if (sqlca.sqlcode != 0) {
      if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
      rc = sqlca.sqlcode;
      if (acid->logging) {
        fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } else {
        fprintf(stderr,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } /* endif */
      sqlerror("acidT: ROLLBACK", &sqlca);
      goto Terror;
    }
    if (acid->logging) {
      gettimeofday(&tv, &tz);
      time(&timeT);
      fprintf(out,"acidT tag: %d, after ROLLBACK: (%us %06uu) %s",
            mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    }
      break;
  }

  acid->l_partkey = l_partkey;
  acid->l_suppkey = l_suppkey;
  acid->l_quantity = l_quantity;
  acid->l_tax = l_tax;
  acid->l_discount = l_discount;
  acid->l_extendedprice = l_extendedprice;
  acid->o_totalprice = o_totalprice;

  rc = 0;
  goto Texit;

 Terror:
    EXEC SQL CLOSE L_CURSOR;
  if (sqlca.sqlcode != 0) sqlerror("acidT: Terror close L_CURSOR failed",
&sqlca);
    EXEC SQL CLOSE O_CURSOR;
  if (sqlca.sqlcode != 0) sqlerror("acidT: Terror close O_CURSOR failed",
&sqlca);
  EXEC SQL rollback work;
  if (sqlca.sqlcode != 0) sqlerror("acidT: ROLLBACK FAILED", &sqlca);

 Texit:
  if (acid->logging) {
    fprintf(out,"\n---------- END of acidT tag: %d ----------\n\n",mypid);
    fflush(out);fclose(out);
  }
  return(rc);
}

/*-----------------------------------------------------------------*/
/*      updateQ                                                    */
/*-----------------------------------------------------------------*/
int updateQ (struct update_struct *us)
{
  FILE *out;
  time_t timeT;
  struct timeval tv;
  struct timezone tz;
  int qnum;
  int rc = 0;
  int i;
  int secs2sleep;
  char buff[256];
  struct acidtype {int logging;} a, *acid;

  EXEC SQL BEGIN DECLARE SECTION;
  double    acctbal;
  double    discount;
  double    price;
  sqlint32      availqty;
  sqlint32      size;
  EXEC SQL END DECLARE SECTION;

  qnum = us->qnum;

  acid = &a;
  acid->logging= 1;

  sprintf(buff, "%s%cupdate.out",getenv("TPCD_TMP_DIR"),del());
  out=fopen(buff,"a");

  gettimeofday(&tv, &tz);
  time(&timeT);
  fprintf(out,"\n---------- START of update ----------\n\n");
  fprintf(out, "update query number: %d, begin transaction time: (%us %06uu)
%s",
        qnum, tv.tv_sec, tv.tv_usec, ctime(&timeT));

  sqlca.sqlcode = 0;
  discount = 0.25;
  price = 5000.50;
  acctbal = 1000.00;
  availqty = 10;
  size = 5;

  for (i=1; i <= 2; i++) {
    gettimeofday(&tv, &tz);
    time(&timeT);
    fprintf(out,"update query number: %d, pass %d, immediately before
UPDATE: (%us %06uu) %s",
          qnum, i, tv.tv_sec, tv.tv_usec, ctime(&timeT));

    switch (qnum)
    {
     case 1:
     {
       EXEC SQL
         UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT + :
discount
         WHERE L_ORDERKEY IN (326,512,928,995);
       if (sqlca.sqlcode != 0) {
         rc = sqlca.sqlcode;
         if (acid->logging)
         {
          fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
         }
         else
         {
          fprintf(stderr,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
         }
         sqlerror("update query number 1", &sqlca);
         goto Uerror;
       }
       discount = discount * (-1);
       secs2sleep = 300;
       break;
     }
     case 2:
     {
       EXEC SQL
         UPDATE TPCD.SUPPLIER set S_ACCTBAL = S_ACCTBAL + :
acctbal
         WHERE S_NAME in
```

```
('Supplier#000000647','Supplier#000000070','Supplier#000000802');
        if (sqlca.sqlcode != 0) {
          rc = sqlca.sqlcode;
          if (acid->logging)
          {
            fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                    qnum, i, sqlca.sqlcode);
          }
          else
          {
            fprintf(stderr,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                    qnum, i, sqlca.sqlcode);
          }
          sqlerror("update query number 2", &sqlca);
          goto Uerror;
        }
        acctbal = acctbal * (-1);
        secs2sleep = 90;
        break;
      }
      case 3:
      {
        EXEC SQL
          UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT + :
discount
          WHERE L_ORDERKEY IN (260930, 402497, 457859, 509889,
58117,
                               538311, 588421, 416167, 97830, 90276);
        if (sqlca.sqlcode != 0) {
          rc = sqlca.sqlcode;
          if (acid->logging)
          {
            fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                    qnum, i, sqlca.sqlcode);
          }
          else
          {
            fprintf(stderr,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                    qnum, i, sqlca.sqlcode);
          }
          sqlerror("update query number 3", &sqlca);
          goto Uerror;
        }
        discount = discount * (-1);
        secs2sleep = 300;
        break;
      }
      case 4:
      {
        if ( i ==1 ) {
          EXEC SQL
            UPDATE TPCD.ORDERS set O_ORDERDATE = O_ORDERDATE
- 6 MONTHS
            WHERE O_ORDERKEY =  67461;
          /* WHERE O_ORDERKEY IN
(22400,28515,34338,46596,67461,92644,98307);*/
        } else {
          EXEC SQL
            UPDATE TPCD.ORDERS set O_ORDERDATE = O_ORDERDATE
+ 6 MONTHS
            WHERE O_ORDERKEY =  67461;
        }
        if (sqlca.sqlcode != 0) {
          rc = sqlca.sqlcode;
          if (acid->logging)
          {
            fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                    qnum, i, sqlca.sqlcode);
          }
          else
          {
            fprintf(stderr,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                    qnum, i, sqlca.sqlcode);
          }
          sqlerror("update query number 4", &sqlca);
          goto Uerror;
        }
        secs2sleep = 300;
        break;
      }
      case 5:
      {
        EXEC SQL
          UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT + :
discount
          WHERE L_ORDERKEY IN (70976,566279,152897,84226,232483);
        if (sqlca.sqlcode != 0) {
          rc = sqlca.sqlcode;
          if (acid->logging)
          {
            fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                    qnum, i, sqlca.sqlcode);
          }
          else
          {
            fprintf(stderr,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                    qnum, i, sqlca.sqlcode);
          }
          sqlerror("update query number 5", &sqlca);
          goto Uerror;
        }
        discount = discount * (-1);
        secs2sleep = 300;
        break;
      }
      case 6:
      {
        EXEC SQL
          UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT + :
discount
          WHERE L_ORDERKEY in
(33,131,161,195,229,230,231,323,353,356);
        if (sqlca.sqlcode != 0) {
          rc = sqlca.sqlcode;
          if (acid->logging)
          {
            fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                    qnum, i, sqlca.sqlcode);
          }
          else
          {
            fprintf(stderr,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                    qnum, i, sqlca.sqlcode);
          }
          sqlerror("update query number 6", &sqlca);
          goto Uerror;
        }
        discount = discount * (-1);
        secs2sleep = 300;
        break;
      }
      case 7:
      {
        EXEC SQL
          UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT + :
discount
          WHERE L_ORDERKEY IN
(562917,410659,16550,398401,157634,429920,45411);
        if (sqlca.sqlcode != 0) {
          rc = sqlca.sqlcode;
          if (acid->logging)
          {
            fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                    qnum, i, sqlca.sqlcode);
          }
          else
          {
            fprintf(stderr,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                    qnum, i, sqlca.sqlcode);
          }
          sqlerror("update query number 7", &sqlca);
          goto Uerror;
```

```
      }
      discount = discount * (-1);
      secs2sleep = 300;
      break;
    }
    case 8:
    {
      EXEC SQL
        UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT + :
discount
        WHERE L_ORDERKEY IN
(129569,343591,270242,254983,98500,28963);
      if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
          fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        else
        {
          fprintf(stderr,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 8", &sqlca);
        goto Uerror;
      }
      discount = discount * (-1);
      secs2sleep = 300;
      break;
    }
    case 9:
    {
      EXEC SQL
        UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT + :
discount
        WHERE L_ORDERKEY IN
(113509,232997,246691,379233,448162,32134);
      if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
          fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        else
        {
          fprintf(stderr,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 9", &sqlca);
        goto Uerror;
      }
      discount = discount * (-1);
      secs2sleep = 300;
      break;
    }
    case 10:
    {
      EXEC SQL
        UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT + :
discount
        WHERE L_ORDERKEY IN
(516487,245411,265799,253025,6914,562020);
      if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
          fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        else
        {
          fprintf(stderr,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 10", &sqlca);
        goto Uerror;
      }
      discount = discount * (-1);
      secs2sleep = 300;
      break;
    }
    case 11:
    {
      EXEC SQL
        UPDATE TPCD.PARTSUPP set PS_AVAILQTY = PS_AVAILQTY + :
availqty
        WHERE PS_PARTKEY IN
(12098,5134,13334,17052,3452,12552,1084,5797);
      if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
          fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        else
        {
          fprintf(stderr,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 11", &sqlca);
        goto Uerror;
      }
      availqty = availqty * (-1);
      secs2sleep = 180;
      break;
    }
    case 12:
    {
      if ( i ==1 ) {
        EXEC SQL
          UPDATE TPCD.LINEITEM set L_RECEIPTDATE =
L_RECEIPTDATE - 3 YEARS
          WHERE L_ORDERKEY IN
(33,70,195,355,677,837,960,962,1028);
      } else {
        EXEC SQL
          UPDATE TPCD.LINEITEM set L_RECEIPTDATE =
L_RECEIPTDATE + 3 YEARS
          WHERE L_ORDERKEY IN
(33,70,195,355,677,837,960,962,1028);
      }
      if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
          fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        else
        {
          fprintf(stderr,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 12", &sqlca);
        goto Uerror;
      }
      secs2sleep = 300;
      break;
    }
    case 13:
    {
      EXEC SQL
        UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT + :
discount
        WHERE L_ORDERKEY IN (263,9476,32355,34854,53445,56901);
      if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
          fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
```

```
                qnum, i, sqlca.sqlcode);
        }
        else
        {
          fprintf(stderr,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 13", &sqlca);
        goto Uerror;
      }
      discount = discount * (-1);
      secs2sleep = 90;
      break;
    }
    case 14:
    {
      EXEC SQL
        UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT + :
discount
        WHERE L_ORDERKEY IN (32,225,326,448,449,483,512);
      if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
          fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        else
        {
          fprintf(stderr,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 14", &sqlca);
        goto Uerror;
      }
      discount = discount * (-1);
      secs2sleep = 180;
      break;
    }
    case 15:
    {
      EXEC SQL
        UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT + :
discount
        WHERE L_ORDERKEY IN (1,4,7,35,135,131300);
      if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
          fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        else
        {
          fprintf(stderr,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 15", &sqlca);
        goto Uerror;
      }
      discount = discount * (-1);
      secs2sleep = 180;
      break;
    }
    case 16:
    {
      EXEC SQL
        UPDATE TPCD.PART set P_SIZE = P_SIZE + :size
        WHERE P_PARTKEY IN (4,7,15,1313);
      if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
          fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        else
        {
          fprintf(stderr,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 16", &sqlca);
        goto Uerror;
      }
      size = size * (-1);
      secs2sleep = 180;
      break;
    }
    case 17:
    {
      EXEC SQL
        UPDATE TPCD.LINEITEM set L_EXTENDEDPRICE =
L_EXTENDEDPRICE + :price
        WHERE L_ORDERKEY IN (4065,110372,165061,265702,87138);
      if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
          fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        else
        {
          fprintf(stderr,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 17", &sqlca);
        goto Uerror;
      }
      price = price * (-1);
      secs2sleep = 90;
      break;
    }
    default:
    {
      fprintf(out,"ERROR: Invalid query number specified %d\n", qnum);
      rc = 1;
      goto Uexit;
    }
    }

    gettimeofday(&tv, &tz);
    time(&timeT);

    if (acid->logging)
      fprintf(out,"update query number: %d, pass %d, after UPDATE: (%us %
06uu) %s",
          qnum, i, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    else
      fprintf(stderr,"update query number: %d, pass %d, after UPDATE: (%us
%06uu) %s",
          qnum, i, tv.tv_sec, tv.tv_usec, ctime(&timeT));

    if ( i == 2 ) {
      gettimeofday(&tv, &tz);
      time(&timeT);
      fprintf(out,"update query number: %d, pass %d, sleeping for %d
seconds: (%us %06uu) %s",
          qnum, i, secs2sleep, tv.tv_sec, tv.tv_usec, ctime(&timeT));
      fflush(out);
      system("touch /tmp/tpcd/update.sync.sleep");
      sleep(secs2sleep);
    }

    gettimeofday(&tv, &tz);
    time(&timeT);
    fprintf(out,"update query number: %d, pass %d, immediately before
COMMIT: (%us %06uu) %s",
        qnum, i, tv.tv_sec, tv.tv_usec, ctime(&timeT));

    EXEC SQL COMMIT;
    if (sqlca.sqlcode != 0) {
      rc = sqlca.sqlcode;
      fprintf(out,"update pass %d, **ERROR** sqlcode = %d\n", i,
sqlca.sqlcode);
```

```
      sqlerror("update: COMMIT", &sqlca);
      goto Uerror;
    }
    gettimeofday(&tv, &tz);
    time(&timeT);
    if (acid->logging)
      fprintf(out,"update query number: %d, pass %d, after COMMIT: (%us %
06uu) %s",
            qnum, i, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    else
      fprintf(stderr,"update query number: %d, pass %d, after COMMIT: (%us
%06uu) %s",
            qnum, i, tv.tv_sec, tv.tv_usec, ctime(&timeT));
  }

  rc = 0;
  goto Uexit;

 Uerror:
  EXEC SQL rollback work;
  if (sqlca.sqlcode != 0) sqlerror("update: ROLLBACK FAILED", &sqlca);
  system("touch /tmp/tpcd/update.sync.sleep");

 Uexit:
  fprintf(out,"\n--------- END of update ----------\n\n");
  fflush(out);fclose(out);
  return(rc);
}

/*-----------------------------------------------------------------*/
/*      connect_to_TM                                              */
/*-----------------------------------------------------------------*/
void connect_to_TM( void )
{
  char *dbname_ptr;
  if ((dbname_ptr = getenv("TPCD_QUAL_DBNAME")) != NULL) {
    fprintf(stderr,"*********** %s ***********\n",dbname_ptr);
    strcpy (dbname, dbname_ptr);
  }

  EXEC SQL CONNECT TO :dbname IN SHARE MODE;
  if (sqlca.sqlcode <  0) {
    fprintf(stderr, "CONNECT TO %s failed SQLCODE = %d\n", dbname,
sqlca.sqlcode);
    exit(-1);
  }
  return;
}

/*-----------------------------------------------------------------*/
/*      disconnect_from_TM                                         */
/*-----------------------------------------------------------------*/
void disconnect_from_TM ( void )
{
  EXEC SQL CONNECT RESET;
  if (sqlca.sqlcode <  0) {
    fprintf(stderr, "DISCONNECT failed SQLCODE = %d\n", sqlca.sqlcode);
    exit(-1);
  }
  return;
}

/*-----------------------------------------------------------------*/
/*      sqlerror                                                   */
/*-----------------------------------------------------------------*/
void sqlerror(char *msg, struct sqlca *psqlca)
{
  FILE *err_fp;


  char  err_fn[256];


  int j,k;

  sprintf(err_fn, "%s%cacid.sqlerrors",getenv("TPCD_TMP_DIR"),del());
  err_fp=fopen(err_fn,"a");
  fprintf(err_fp,"acid: sqlcode: %4d %s\n", psqlca->sqlcode, msg);
  fprintf(stderr,"acid: sqlcode: %4d %s\n", psqlca->sqlcode, msg);
  fflush(stderr);
  if (psqlca->sqlerrmc[0] != ' ' || psqlca->sqlerrmc[1] != ' ') {
    fprintf(err_fp,"acid: slerrmc:  ");
    for(j = 0; j < 5; j++)
    {
      for(k = 0; k < 14; k++) fprintf(err_fp,"%x ", psqlca->sqlerrmc[j*10+k]);
      fprintf(err_fp,"    ");
      for(k = 0; k < 14; k++) fprintf(err_fp,"%c",  psqlca->sqlerrmc[j*10+k]);
      fprintf(err_fp,"\n");
      if (j < 4) fprintf(err_fp,"          ");
    }
  }

  fprintf(err_fp,"acid: sqlerrp:  ");
  for(j = 0; j < 8; j++)    fprintf(err_fp,"%c", psqlca->sqlerrp[j]);
  fprintf(err_fp,"\n");

  fprintf(err_fp,"acid: sqlerrd: ");
  for(j = 0; j < 6; j++)    fprintf(err_fp," %d", psqlca->sqlerrd[j]);
  fprintf(err_fp,"\n");

  if (psqlca->sqlwarn[0] != ' ') {
    fprintf(err_fp,"acid: sqlwarn: ");
    for(j = 0; j < 8; j++)    fprintf(err_fp,"%c ", psqlca->sqlwarn[j]);
    fprintf(err_fp,"\n");
  }

  fprintf(err_fp,"\n");
  fflush(err_fp);fclose(err_fp);
}

#ifdef SQLWINT
void sleep(int sec)
{
 Sleep(sec * 1000);
}
#endif


char del(void)
{
#ifdef SQLWINT
  return '\\';
#else
  return '/';
#endif
}

#if defined(SQLPTX) || defined(SQLWINT) || defined(SQLSUN) || defined
(Linux)
/* added fot 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 */
```

mainacid.c

```
/*****************************************************************
**********/
/*      File: mainacid.c                                          */
/*****************************************************************
**********/

/*
** changes:
```

```
**
**   gav  - Feb 25, 1998:    port to NT.
**   kal  - Oct 06, 1998:    add summary table acid query Txn3.
**
*/

#include "acid.h"
#include <time.h>
#include <stdio.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 <errno.h>
#endif

#include <fcntl.h>

#define ACID_QUERY       1
#define ACID_TRANS       2
#define MULTI_ACID_TRANS 3
#define UPDATE           4
#define AST_ACID_QUERY   5
#define PERMS 0644

#define WRITE(fd, bp, bl, id) \
            if (1) { \
                        int wr = write(fd, bp, bl); \
                        if (bl != wr) \
                                    fprintf(stderr, "ERROR on write, id = %d,
errno = %d, req = %d, ret = %d\n", \
                                                id, errno, bl, wr); \
                        if (-1 == fsync(fd)) \
                                    fprintf(stderr, "ERROR on fsync, id = %d,
errno = %d\n", id, errno); \
            }

extern int acidQ (struct acidQ_struct *);
extern int ast_acidQ (struct acidQ_struct *);
extern int acidT (struct acidT_struct *);
extern void connect_to_TM( void );
extern void disconnect_from_TM( void );
/*extern void sqlerror(char * , struct sqlca *);*/


int acidQ_sample (struct acidQ_struct *);
int ast_acidQ_sample (struct acidQ_struct *);
int acidT_sample (struct acidT_struct *);
int rand_integer (int, int);
void print_time(void);
char *current_tmstmp();

long o_key, l_key, delta;
int  termination, tag;

#ifdef SQLWINT
  struct timeb timer1;
#else
  struct timeval timer1;
  int    procid;
#endif

/*-----------------------------------------------------*/
/*            main                                     */
/*-----------------------------------------------------*/
int main (int argc, char *argv[])
{
  char   Abuffer[1024];
  char   buf[512];
  int    buf_len;
  char   infile[256];
  char   ratefile[256];
  char   outfile[256];
  struct acidQ_struct *p_acidQ_s; /* will be used for ast acid query or acid
query */
  struct acidT_struct *p_acidT_s;
  struct update_struct *p_update_s;
#ifdef SQLWINT
  FILE *log_file;
#else
  int log_fd;
#endif
  int rate_fd;
  FILE *log;
  FILE *in_fp;
  char buff[256];


  int    t_or_q;
  int    ret, i;

  memset ((void *) Abuffer, 0, 1024);
  memset ((void *) buf, 0, 512);
  t_or_q = termination = tag = o_key = l_key = delta = 0;
  switch (argc) {
   case 7:
     delta = (atoi(argv[6]));
   case 6:
     l_key = (atoi(argv[5]));
   case 5:
     o_key = (atoi(argv[4]));
   case 4:
     tag = (atoi(argv[3]));
   case 3:
     termination = (atoi(argv[2]));  /* = commit/rollback and        */
                                     /*   wait/nowait for ACID_TRANS */
                                     /* = no of tranactions for      */
                                     /*   MULTI_ACID_TRANS           */
   case 2:
     t_or_q= (atoi(argv[1]));
     break;
  }
  if (t_or_q < 1 || t_or_q > 5) {
    printf("*** ERROR *** mainacid not invoked properly\n");
    printf("Usage:  mainacid acidType termination tag o_key l_key delta\n");
    printf("where:\n");
    printf("\tacidType    - 1 (Acid Query)\n");
    printf("\t            2 (Acid Transaction)\n");
    printf("\t            3 (Multiple Acid Transactions)\n");
    printf("\t            4 (Updates)\n");
    printf("\t            5 (Summary Table Acid Query)\n");
    printf("\ttermination - 0 (commit no wait (default))\n");
    printf("\t            1 (commit with wait)\n");
    printf("\t            2 (rollback no wait)\n");
    printf("\t            3 (rollback with wait)\n");
    printf("\ttag         - unique numerical identifier used at end of the\n");
    printf("\t            name of the output file /tmp/tpcd/acidX.{tag}\n");
    printf("\to_key       - default is random generation\n");
    printf("\tl_key       - default is random generation\n");
    printf("\tdelta       - default is random generation\n\n");
    exit(1);
  }

#ifdef SQLWINT
  srand( (unsigned)time(NULL));
#else
  procid = getpid();
  srand(procid);
  srandom(procid);
#endif

  connect_to_TM();                    /* Start using database */

  switch (t_or_q) {
   case ACID_QUERY:
  {
    printf("Acid query called at ");
    print_time();
    p_acidQ_s = (struct acidQ_struct *)Abuffer;
    acidQ_sample(p_acidQ_s);
    ret = acidQ(p_acidQ_s);
    if (ret != 0) {
      fprintf(stderr,"Acid query had non zero return code of %d\n",ret);
    }

    sprintf(buff,"%s%cmainacid.log.Q",getenv("TPCD_TMP_DIR"),del());
    log=fopen(buff,"a");
```

```
    fprintf(log,"{tag: %d, o_key: %d, l_extendedprice: %0.2f, %s} %d\n",
            p_acidQ_s->tag, p_acidQ_s->o_key,
            p_acidQ_s->l_extendedprice, current_tmstmp(), ret);
    fflush(log);fclose(log);
    break;
  }
  case AST_ACID_QUERY:
  {
    printf("AST Acid query called at ");
    print_time();
    p_acidQ_s = (struct acidQ_struct *)Abuffer;
    ast_acidQ_sample(p_acidQ_s);
    ret = ast_acidQ(p_acidQ_s);
    if (ret != 0) {
      fprintf(stderr,"AST Acid query had non zero return code of %d\n",ret);
    }

    sprintf(buff,"%s%cmainacid.log.Q",getenv("TPCD_TMP_DIR"),del());
    log=fopen(buff,"a");

    fprintf(log,"{tag: %d, l_extendedprice: %0.2f, %s} %d\n",
            p_acidQ_s->tag, p_acidQ_s->l_extendedprice,
            current_tmstmp(), ret);
    fflush(log);fclose(log);
    break;
  }
  case ACID_TRANS:
  {
    printf("Acid Transaction called at ");
    print_time();
    p_acidT_s = (struct acidT_struct *)Abuffer;
    acidT_sample(p_acidT_s);
    ret = acidT(p_acidT_s);

    if (ret != 0) {
      fprintf(stderr,"Acid transaction had non zero return code of %d\n",ret);
      sprintf(buff,"%s%cmainacid.log.T.errors",getenv("TPCD_TMP_DIR"),
del());
      log=fopen(buff,"ac");
    }
    else
    {
      sprintf(buff,"%s%cmainacid.log.T",getenv("TPCD_TMP_DIR"),del());
      log=fopen(buff,"ac");
    }

    fprintf(log,"term %d tag %d o_key %d l_key %d delta %d l_partkey %d
l_suppkey %d l_quantity %0.2f l_tax %0.2f l_discount %0.2f l_extendedprice
%0.2f o_totalprice %0.2f timestamp %s return_code %d\n",
            p_acidT_s->termination,
            p_acidT_s->tag,
            p_acidT_s->o_key,
            p_acidT_s->l_key,
            p_acidT_s->delta,
            p_acidT_s->l_partkey,
            p_acidT_s->l_suppkey,
            p_acidT_s->l_quantity,
            p_acidT_s->l_tax,
            p_acidT_s->l_discount,
            p_acidT_s->l_extendedprice,
            p_acidT_s->o_totalprice,
            current_tmstmp(), ret);
    fflush(log);fclose(log);
    break;
  }
  case MULTI_ACID_TRANS:
  {
    sprintf(ratefile,"%s%crate.%d",getenv("TPCD_TMP_DIR"),del(),tag);


#ifdef SQLWINT
    if ((rate_fd = _open(ratefile, _O_CREAT|_O_WRONLY|_O_TRUNC|
_O_APPEND, _S_IWRITE)) == -1) {
#else
    if ((rate_fd = open(ratefile, O_CREAT|O_WRONLY|O_TRUNC|O_SYNC|
O_APPEND, PERMS)) == -1) {
#endif
      fprintf(stderr,"Error! Cannot open/create file %s, mode %03o\n",
ratefile,PERMS);
      goto myexit;
    }
#ifdef SQLWINT
    ftime(&timer1);
    sprintf(buf,"-- mainacid (Stream %d) called at %us %06uu\n\0", tag,
timer1.time  , timer1.millitm);
#else
    gettimeofday(&timer1,0);
    sprintf(buf,"-- mainacid (Stream %d) called at %us %06uu\n\0", tag,
timer1.tv_sec, timer1.tv_usec);
#endif

    buf_len = strlen(buf);
    WRITE(rate_fd, buf, buf_len, 1);

    sprintf(infile,"%d.in.%d",termination, tag);
    if ((in_fp = fopen(infile, "r")) == NULL)
    {
      fprintf(stderr,"ERROR input file %s could not be read!!\n",infile);
      goto myexit;
    }
    sprintf(outfile,"%s%cmainacid.log.T.%d",getenv("TPCD_TMP_DIR"),del
(),tag);

    /*
    ** in the following code we make sure that the log file
    ** is opened without nasty operating-system
    ** file caching.  This is OS-specific stuff.
    ** on AIX, this is achieved with the right options on
    ** the "open()" command.  on NT, it can only be done
    ** with FILE structures, and it must be opened with
    ** the "c" option.  on NT, only fflush() will cause
    ** the data to be actually written to disk!
    ** fclose() is not good enough.
    */
#ifdef SQLWINT
    if ((log_file = fopen(outfile,"ac"))  == NULL) {
#else
    if ((log_fd = open(outfile, O_CREAT|O_WRONLY|O_TRUNC|O_SYNC|
O_APPEND, PERMS)) == -1) {
#endif
      fprintf(stderr,"Error! Cannot open/create file %s, mode %03o\n", outfile,
PERMS);
      goto myexit;
    }
    p_acidT_s = (struct acidT_struct *)Abuffer;
    for (i=1; i <= termination; i++)
    {
      printf("Stream %d running acid transaction %d/%d\n",tag,i,termination);
      fscanf(in_fp,"%d %d",&o_key, &l_key);
      acidT_sample(p_acidT_s);
      p_acidT_s->termination = 0;  /* do commit, no wait */
      p_acidT_s->logging = 0;

#ifdef SQLWINT
      ftime(&timer1);
      sprintf(buf,"Acid transaction called at %us %06uu (Stream %d) \0",
            timer1.time  , timer1.millitm, tag);
#else
      gettimeofday(&timer1,0);
      sprintf(buf,"Acid transaction called at %us %06uu (Stream %d) \0",
            timer1.tv_sec, timer1.tv_usec, tag);
#endif
      buf_len = strlen(buf);
      WRITE(rate_fd, buf, buf_len, 2);
      ret = acidT(p_acidT_s);


#ifdef SQLWINT
      ftime(&timer1);
      sprintf(buf,"(%us %06uu)\n\0", timer1.time  , timer1.millitm);
#else
      gettimeofday(&timer1,0);
      sprintf(buf,"(%us %06uu)\n\0", timer1.tv_sec, timer1.tv_usec);
#endif
      buf_len = strlen(buf);
      WRITE(rate_fd, buf, buf_len, 3);

      sprintf(buf,"term %d tag %d o_key %d l_key %d delta %d l_partkey %d
l_suppkey %d l_quantity %0.2f l_tax %0.2f l_discount %0.2f l_extendedprice
%0.2f o_totalprice %0.2f timestamp %s return_code %d\n\0",
            p_acidT_s->termination,
            p_acidT_s->tag,
            p_acidT_s->o_key,
```

```
            p_acidT_s->l_key,
            p_acidT_s->delta,
            p_acidT_s->l_partkey,
            p_acidT_s->l_suppkey,
            p_acidT_s->l_quantity,
            p_acidT_s->l_tax,
            p_acidT_s->l_discount,
            p_acidT_s->l_extendedprice,
            p_acidT_s->o_totalprice,
            current_tmstmp(), ret);
        if (ret != 0) {
          fprintf(stderr,"Acid transaction had non zero return code of %d\n",ret);
          sprintf(outfile,"%s%cmainacid.log.T.errors",getenv
("TPCD_TMP_DIR"),del());
          log=fopen(outfile,"a");
          fprintf(log,"%s",buf);
          fflush(log);fclose(log);
          fprintf(stderr,"Terminating execution! %d transactions not executed for
stream %d\n",
                (termination-i),tag);
          break; /* do not process any more transactions */
        } else {
          buf_len = strlen(buf);
          #ifdef SQLWINT
            fprintf(log_file, "%s",buf);
            fflush(log_file);
          #else
            WRITE(log_fd, buf, buf_len, 4);
          #endif

        }
      }
      #ifdef SQLWINT
        fclose(log_file);
      #else
        close(rate_fd);
      #endif
      fflush(in_fp);fclose(in_fp);
      break;
    }
    case UPDATE:
    {

      printf("Update called at ");
      print_time();
      p_update_s = (struct update_struct *)Abuffer;
      p_update_s->qnum = termination;
      ret = updateQ(p_update_s);
      if (ret != 0) {
        fprintf(stderr,"Update had non zero return code of %d\n",ret);
      }
      sprintf(outfile,"%s%cmainacid.log.U",getenv("TPCD_TMP_DIR"),del());
      log=fopen(outfile,"a");

      fprintf(log,"{query number: %d, %s} %d\n",
            p_update_s->qnum, current_tmstmp(), ret);
      fflush(log);fclose(log);
      break;
    }
    }
  myexit:
    disconnect_from_TM();

    return 0;

} /* end of main          */

/*-----------------------------------------------------*/
/*            acidQ sample                             */
/*-----------------------------------------------------*/
int acidQ_sample (struct acidQ_struct *acid)
{
  acid->tag = tag;
  if (o_key) acid->o_key = o_key;
  else acid->o_key = rand_integer(1,150000);

  return 0;
}

/*-----------------------------------------------------*/
/*            ast acidQ sample                         */
/*-----------------------------------------------------*/
int ast_acidQ_sample (struct acidQ_struct *acid)
{
  acid->tag = tag;
  return 0;
}

/*-----------------------------------------------------*/
/*            acidT sample                             */
/*-----------------------------------------------------*/
int acidT_sample (struct acidT_struct *acid)
{
  acid->termination = termination;
  acid->tag = tag;
  acid->logging = 1;
  if (o_key) acid->o_key = o_key;
  else acid->o_key = rand_integer(1,150000);
  if (l_key) acid->l_key = l_key;
  else acid->l_key = rand_integer(1,7);
  if (delta) acid->delta = delta;
  else acid->delta = rand_integer(1,100);

  return 0;
}

/*-----------------------------------------------------------*/
/*   rand_integer                                            */
/*-----------------------------------------------------------*/
int rand_integer (int val_lo, int val_hi)
{
#ifdef SQLWINT
  return((rand() % (val_hi-val_lo+1)) + val_lo);
#else
  return((random() % (val_hi-val_lo+1)) + val_lo);
#endif
}

/*-------------------------------------------------------------------*/
/*     current_tmstmp                                                */
/*-------------------------------------------------------------------*/
char *current_tmstmp()
{
 time_t   tlong;
 struct tm *lt;
 static char tstring[TSLEN];
 time(&tlong);
 lt = localtime(&tlong);
 sprintf(tstring, "%04d-%02d-%02d-%02d.%02d.%02d",
        lt->tm_year+1900, lt->tm_mon+1, lt->tm_mday,
        lt->tm_hour,      lt->tm_min,   lt->tm_sec);
 return (tstring);
}

/*-------------------------------------------------------------------*/
/*     print_time                                                    */
/*-------------------------------------------------------------------*/
void print_time()
{
#ifdef SQLWINT
    ftime(&timer1);
    printf("TIME = %us %06uu\n\0", timer1.time  , timer1.millitm);
#else
    gettimeofday(&timer1,0);
    printf("TIME = %us %06uu\n\0", timer1.tv_sec, timer1.tv_usec);
#endif
}
```

Appendix F: Price Quotes

MAR 04 2005 04:02 PM TO 916509330193 P.02/02

March 4, 2005

Doug Rady
SGI Corporation,
Mountain View, CA 94043.

Dear Mr. Rady,

The table shown below lists the U.S. pricing for DB2 Universal Database Enterprise Server Edition product that has been used in TPC-H Benchmark test.

All prices shown are in U.S. Dollars.

DB2 Enterprise Server Edition (ESE)	Qty	Reference Price per unit	Total Reference price
SW License & 1 year Maintenance	8	22,260	178,080
SW Maintenance Renewal - 1 year	16	1,060	16,960
		Sub-total reference price for DB2 ESE:	**195,040**

DB2 Database Partitioning Feature (DPF)	Qty	Reference Price per unit	Total Reference price
SW License & 1 year Maintenance	8	6,686	53,488
SW Maintenance Renewal - 1 year	16	318	5,088
		Sub-total reference price for DB2 DPF:	**58,576**
		TOTAL REFERENCE PRICE:	**253,616**

This quote is valid for 90 days.

If I can be of any further assistance, please contact me at 914-766-1491or spang@us.ibm.com.

Yours Truly,

Bernard Spang
Director, Database Market Management
IBM DB2 Information Management Software
Route 100
Somers, NY 10589

** TOTAL PAGE.02 *