TPC Benchmark™ H
Full Disclosure Report

Sun Microsystems Sun Fire™ X4500
Using IBM DB2 9.1

Submitted for Review
Report Date October 12, 2007

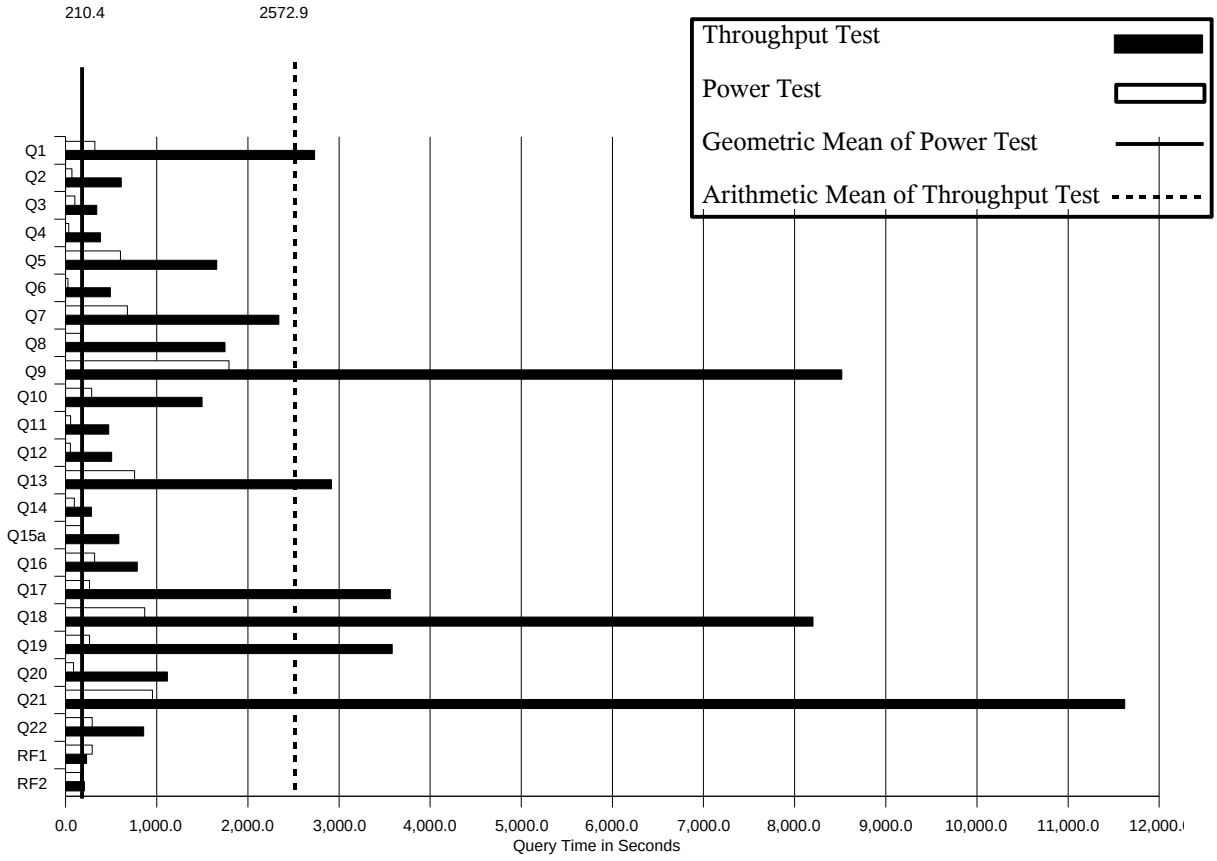
First Printing

901 San Antonio Road, Palo Alto, California 94303 U.S.A.

Sun microsystems / IBM	**Sun Fire™ X4500 Server with IBM DB2 9.1**	**TPC-H Rev. 2.6** **October 12, 2007**

Total System Cost	Composite Query per Hour Metric	Price/Performance
$1,136,536 USD	**38,672.4** QphH@3000GB	**$29.39 USD** $/QphH@3000GB

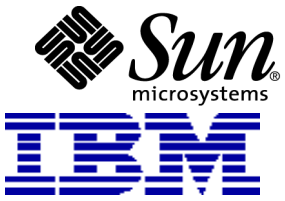
Database Size	Database Manager	Operating System	Other Software	Availability Date
3000GB	IBM DB2 9.1	Solaris 10		October 12, 2007



Database Load Time = 3:39	Load Includes Backup: N	Total Data Storage/Database Size=74.5
RAID (Base tables): N	RAID (Base tables and auxiliary data structures): N	RAID (All): Y

System Configuration: 10 Sun Fire™ X4500 Servers, each server with:

Processors:	2 AMD Dual Core Opteron Model 285 on 2 chips, 4 cores, 4 threads
Memory:	16 GB memory
Disks:	48 465.8 GB SATA Disk Drives @ 7200rpm
Total Storage:	218.3 TB (in this calculation one TB is defined as 1024*1024*1024*1024 bytes)



Sun Fire™ X4500 Server with IBM DB2 9.1

TPC-H Rev. 2.6

October 12, 2007

Description	Part Number	Source	Unit Price	Qty	Ext. Price	3 Yr. Maint.
Server Hardware						
Sun Fire X4500 x64 Server	A76-PGZ2-16G-HMH	1	34,995	10	349,950	86,760
Sun Rack 1000-38	SR2-21038A	1	3,075	2	6,150	1,008
CCB/Pwr strips, 1000-38	9233A-2	1	500	4	2,000	
MPS PWR 30A/single phase	9241A-2	1	1,000	4	4,000	
Pwr Jmpr cbls rack 1.0mx1	X9237-1-A	1	6	22	132	
Belkin 8' CAT5e Patch Cable, Snagless Molded, RJ45, Gray	443112	4	8	30	235	
APC Smart-UPS RM 3000VA USB	592697	4	1,332	5	6,662	
3 Year DOP Warranty w/ADH $701-$2000	1135230	4	284	5		1,420
NETGEAR JGS524 24-port Gigabit Ethernet Switch	652859	4	270	2	540	
NETGEAR 3 Year ProSupport Maintenance Contract - OnCall 24x7 - Category 1	867761	4	132	2		264
Server Hardware Subtotal					369,669	89,188
Server Software						
Solaris 10 Media		3	30	1	30	0
Sun Studio 11 C/C++ Media		1	10	1	10	3,060
DB2 ESE 9 Value Unit Lic/1 year Maintenance		2	278.52	2000	557,040	0
DB2 ESE SW Maintenance Renewal – 1Year		2	13.27	4000		53,080
DB2 DPF License & 1 year maintenance		2	74.32	2000	148,640	
DB2 DPF Maintenance Renewal – 1Year		2	3.54	4000		14,160
Server Software Subtotal					705,720	70,300
Sun Volume Discounts (18%) and Support Prepayment (35%)		1			-66,540	-31,790
			Total		1,008,848	127,698
Service for all Sun products is from Sun Microsystems, Inc. and is based on SunSpectrum Instant Upgrade Gold 7x24			**3 Yr. Cost**		1,136,546	
Service for DB2 products is from IBM Corp.			**QphH@3000GB**		38,672	
			$/QphH@3000GB		$29.39	

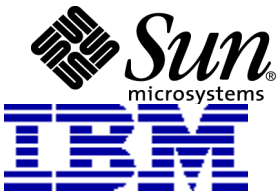
Notes (Source):

1. Sun Store U.S. (http://store.sun.com)
2. IBM Corp.
3. Download from SunSolve
4. cdw.com

All prices are in USD

Audited by: François Raab, InfoSizing, Inc. (www.sizing.com)

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



Sun Fire™ 25K Server with IBM DB2 9.1

TPC-H Rev. 2.6

October 12, 2007

Numerical Quantities

Measurement Results:

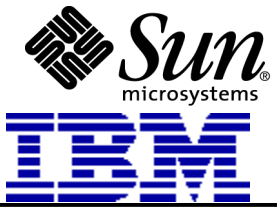
Database Scale Factor	= 3000GB
Total Data Storage / Database Size	= 74.5
Start of database load time	= 09-21-2007 21:35:28
End of database load time	= 09-22-2007 01:14:55
Database Load Time	= 3:39
Query Streams for Throughput Test	= 8
TPC-H Power	= 51,320.0
TPC-H Throughput	= 29,141.8
TPC-H Composite Query-per-Hour Rating (QphH@3000GB)	= 38,672.4
Total System Price Over 3 Years	= $1,136,536
TPC-H Price/Performance Metric ($/QphH@3000GB)	= $29.39

Measurement Intervals:

Measurement Interval in Throughput Test (Ts)	= 65,226 seconds

Duration of Stream Execution:

		Date and Time Stamps						Duration
Stream Id	Seed Used	Query Start		RF1 Start		RF2 Start		Queries
		Query End		RF1 End		RF2 End		RFs
00	923011455	9/22/07	1:54:31	9/22/07	1:49:41	9/22/07	4:12:09	2:17:38
		9/22/07	4:12:09	9/22/07	1:54:31	9/22/07	4:14:56	0:07:37
01	923011456	9/22/07	4:15:00	9/22/07	4:15:15	9/22/07	21:28:04	17:00:12
		9/22/07	21:15:11	9/22/07	21:28:04	9/22/07	21:31:11	0:15:44
02	923011457	9/22/07	4:15:00	9/22/07	21:31:11	9/22/07	21:34:52	16:56:35
		9/22/07	21:11:34	9/22/07	21:34:52	9/22/07	21:38:14	0:07:03
03	923011458	9/22/07	4:15:00	9/22/07	21:38:14	9/22/07	21:41:55	17:09:18
		9/22/07	21:24:18	9/22/07	21:41:55	9/22/07	21:45:13	0:06:59
04	923011459	9/22/07	4:15:00	9/22/07	21:45:13	9/22/07	21:48:50	16:33:31
		9/22/07	20:48:31	9/22/07	21:48:50	9/22/07	21:52:12	0:06:59
05	923011460	9/22/07	4:15:00	9/22/07	21:52:12	9/22/07	21:55:56	16:58:56
		9/22/07	21:13:56	9/22/07	21:55:56	9/22/07	21:59:57	0:07:45
06	923011461	9/22/07	4:15:00	9/22/07	21:59:57	9/22/07	22:03:43	16:44:05
		9/22/07	20:59:05	9/22/07	22:03:43	9/22/07	22:07:25	0:07:28
07	923011462	9/22/07	4:15:00	9/22/07	22:07:25	9/22/07	22:11:08	16:08:54
		9/22/07	20:23:54	9/22/07	22:11:08	9/22/07	22:14:42	0:07:17
08	923011463	9/22/07	4:15:00	9/22/07	22:14:42	9/22/07	22:18:22	17:03:44
		9/22/07	21:18:44	9/22/07	22:18:22	9/22/07	22:22:06	0:07:24



Sun Fire™ 25K Server with IBM DB2 9.1

TPC-H Rev. 2.6

October 12, 2007

TPC-H Timing Interval (in seconds)

Stream ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
00	322.1	69.5	102.5	34.8	601.9	26.3	677.8	162.0
01	1,026.3	739.9	520.3	84.9	1,635.0	407.4	2,488.9	2,332.5
02	4,422.0	785.4	458.4	427.3	1,636.2	661.9	1,950.3	1,970.5
03	4,745.4	856.8	104.5	404.1	1,812.8	461.1	2,772.6	1,991.5
04	4,588.3	468.6	270.4	382.9	1,868.6	675.2	3,109.6	1,096.4
05	1,967.5	623.8	380.0	470.9	1,717.4	717.0	2,826.2	2,142.4
06	1,926.4	589.3	498.9	296.6	1,965.8	541.6	1,973.1	1,804.0
07	2,149.4	783.8	449.2	496.3	1,937.2	675.9	2,769.0	2,113.3
08	3,446.0	584.4	301.9	272.4	1,751.0	266.0	2,486.2	2,117.4
Minimum	322.1	69.5	102.5	34.8	601.9	26.3	677.8	162.0
Average	2,732.6	611.3	342.9	318.9	1,658.4	492.5	2,339.3	1,747.8
Maximum	4,745.4	856.8	520.3	496.3	1,965.8	717.0	3,109.6	2,332.5

Stream ID	Q9	Q10	Q11	Q12	Q13	Q14	Q15a	Q16
00	1,794.6	285.2	54.2	52.0	758.5	95.5	176.6	317.6
01	10,800.8	1,684.8	935.3	392.4	3,628.1	321.2	768.9	998.6
02	10,709.6	1,677.7	746.0	658.8	4,041.0	291.0	517.0	760.5
03	8,037.0	1,731.5	450.2	55.1	2,294.1	324.6	866.3	445.8
04	8,566.5	1,370.4	529.9	627.0	2,876.2	307.2	563.2	1,065.7
05	9,122.2	1,695.6	502.8	879.7	3,532.6	313.0	662.0	775.9
06	11,079.6	1,799.9	584.5	445.6	2,790.8	248.9	581.2	1,042.2
07	7,853.6	1,797.5	389.7	756.6	2,402.1	313.8	525.9	831.5
08	8,698.5	1,423.7	69.3	683.2	3,924.0	333.9	611.9	850.4
Minimum	1,794.6	285.2	54.2	52.0	758.5	95.5	176.6	317.6
Average	8,518.0	1,496.3	473.5	505.6	2,916.4	283.2	585.9	787.6
Maximum	11,079.6	1,799.9	935.3	879.7	4,041.0	333.9	866.3	1,065.7

Stream ID	Q17	Q18	Q19	Q20	Q21	Q22	RF1	RF2
00	261.7	869.8	262.2	86.6	954.3	291.8	290.8	167.2
01	2,771.7	10,225.0	4,600.5	959.5	13,050.3	839.7	226.0	187.7
02	4,249.8	8,631.3	4,131.9	1,766.6	9,697.2	804.6	221.1	201.3
03	3,707.4	9,709.3	3,899.9	988.9	15,136.2	963.0	221.0	197.8
04	2,986.4	8,999.7	3,816.2	879.7	13,712.5	850.5	217.7	202.1
05	4,978.2	9,469.2	2,909.1	1,497.5	12,877.5	1,075.5	224.0	240.7
06	4,673.4	9,280.4	3,643.4	1,489.5	12,356.8	633.4	225.9	221.7
07	3,469.2	9,284.3	4,731.6	1,438.7	11,871.7	1,093.2	223.0	214.5
08	4,961.5	7,339.7	4,258.7	944.7	14,933.8	1,165.7	219.9	223.9
Minimum	261.7	869.8	262.2	86.6	954.3	291.8	217.7	167.2
Average	3,562.1	8,201.0	3,583.7	1,116.9	11,621.1	857.5	229.9	206.3
Maximum	4,978.2	10,225.0	4,731.6	1,766.6	15,136.2	1,165.7	290.8	240.7

INFO SIZING

TPC TRANSACTION PROCESSING PERFORMANCE COUNCIL CERTIFIED AUDITOR

Benchmark Sponsor:

Brad Carlile
Director, Enterprise Benchmarking
Sun Microsystems, Inc.
3295 N.W. 211th Terrace
Hillsboro OR, 97124

Haider Rizvi
Senior Manager, DB2 Data
Warehousing Performance
IBM Canada Ltd.
8200 Warden Ave,
Markham, Ontario L6G 1C7

October 10, 2007

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

Platform: **Sun Fire™ X4500 Server, 10-node Cluster**
Database Manager: **IBM DB2 9.1**
Operating System: **Solaris 10**

The results were:

CPU (Speed)	Memory	Disks	**QphH @3000GB**
10-node Cluster of Sun Fire™ X4500 Servers (each node with)			
2 x AMD Dual Core Opteron Model 285 (2.6 GHz)	16 GB Main	48 x 465.8 GB 7200rpm SATA	**38,672.4**

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

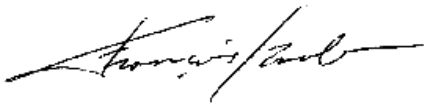
- The database records were defined with the proper layout and size
- The database population was generated using DBGEN
- The database was properly scaled to 3,000GB and populated accordingly
- The compliance of the database auxiliary data structures was verified
- The database load time was correctly measured and reported

- The required ACID properties were verified and met
- The query input variables were generated by QGEN
- The query text was produced using minor modifications and one query variant
- The execution of the queries against the SF1 database produced compliant answers
- A compliant implementation specific layer was used to drive the tests
- The throughput tests involved 8 query streams
- The ratio between the longest and the shortest query was such that no query timing was adjusted
- The execution times for queries and refresh functions were correctly measured and reported
- The repeatability of the measured results was verified
- The required amount of database log was configured
- The system pricing was verified for major components and maintenance
- The major pages from the FDR were verified for accuracy

Additional Audit Notes:

none.

Respectfully Yours,



François Raab
President

125 West Monroe Street • Colorado Springs, CO 80907 • 719-473-7555 • www.sizing.com

Table of Contents

1. General Items 12
   1.1 Benchmark Sponsor 12
   1.2 Parameter Settings 12
   1.3 Configuration Diagram 13
2. Clause 1 Logical Database Design 15
   2.1 Database Definition Statements 15
   2.2 Physical Organization 15
   2.3 Horizontal Partitioning 15
   2.4 Replication 15
3. Clause 2 Queries and Refresh Functions 16
   3.1 Query Language 16
   3.2 Verifying Method for Random Number Generation 16
   3.3 Generating Values for Substitution Parameters 16
   3.4 Query Text and Output Data from Qualification Database 16
   3.5 Query Substitution Parameters and Seeds Used 16
   3.6 Query Isolation Level 17
   3.7 Source Code of Refresh Functions 17
4. Clause 3 Database System Properties 18
   4.1 ACID Properties 18
   4.2 Atomicity 18
      4.2.1 Completed Transaction 18
      4.2.2 Aborted Transaction 18
   4.3 Consistency 18
      4.3.1 Consistency Test 19
   4.4 Isolation 19
      4.4.1 Read-Write Conflict with Commit 19
      4.4.2 Read-Write Conflict with Rollback 19
      4.4.3 Write-Write Conflict with Commit 19
      4.4.4 Write-Write Conflict with Rollback 20
      4.4.5 Concurrent Progress of Read and Write Transactions 20
      4.4.6 Read-Only Query Conflict with Update Transaction 20
   4.5 Durability 21
      4.5.1 Failure of a Durable Medium 21
      4.5.2 System Crash 21
      4.5.3 Memory Failure 21
5. Clause 4 Scaling and Database Population 22
   5.1 Ending Cardinality of Tables 22
   5.2 Distribution of Tables and Logs Across Media 22
   5.3 Database partition/replication mapping 22
   5.4 RAID Feature 23
   5.5 Modifications to the DBGEN 23
   5.6 Database Load Time 23
   5.7 Data Storage Ratio 23
   5.8 Database Load Mechanism Details and Illustration 24
   5.9 Qualification Database Configuration 24
   5.10 Dataset verification 24

5.11 Referential Integrity 24
6. Clause 5 Performance Metrics and Execution Rules 25
6.1 System Activity Between Load and Performance Tests 25
6.2 Steps in the Power Test 25
6.3 Timing Intervals for Each Query and Refresh Functions 25
6.4 Number of Streams for the Throughput Test 25
6.5 Start and End Date/Times for Each Query Stream 25
6.6 Total Elapsed Time of the Measurement Interval 26
6.7 Refresh Function Start Date/Time and Finish Date/Time 26
6.8 Timing Intervals for Each Query and Each Refresh Function for Each Stream 26
6.9 Performance Metrics 26
6.10 The Performance Metric and Numerical Quantities from Both Runs 26
6.11 System Activity Between Performance Tests 26
7. Clause 6 SUT and Driver Implementation 27
7.1 Driver 27
7.2 Implementation-Specific Layer 27
7.3 Profile-Directed Optimization 28
8. Clause 7 Pricing 29
8.1 Hardware and Software Used 29
8.2 Total Three Year Price 29
8.3 Availability Date 29
9. Auditor's Information and Attestation Letter 30
Appendix A. Solaris 10 and DB2 Parameters 31
Appendix B. Programs and Scripts 32
Appendix C. Query Text and Query Output 85
Appendix D. Seed and Query Substitution Parameters 104
Appendix E. Implementation-Specific Layer/Driver Code 108
Appendix F. Pricing information 256

TPC Benchmark H Overview

The TPC BenchmarkTM H (TPC-H) is a Decision Support benchmark. It is a suite of business-oriented ad-hoc queries and concurrent 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 sets 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
- Are far more complex than most OLTP transactions
- Include a rich breadth of operators and selectivity constraints
- Generate intensive activity on the part of the database server component of the system under test
- Are executed against a database complying to specific population and scaling requirements
- Are implemented with constraints derived from staying closely synchronized with an on-line production database

1. General Items

1.1 Benchmark Sponsor

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

Sun Microsystems, Inc. and IBM Corp. are the sponsors of 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*

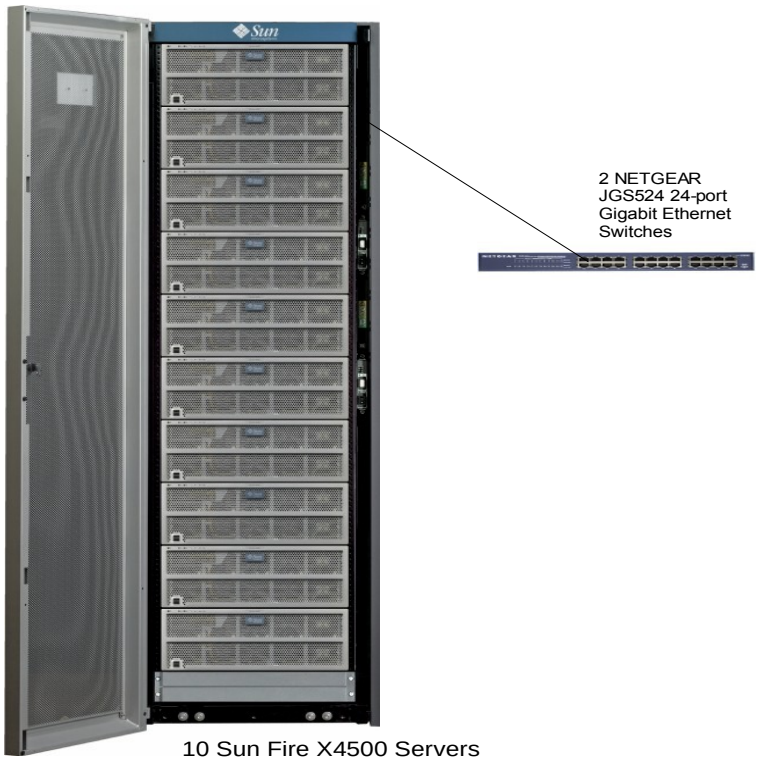
Appendix A contains the Solaris and DB2 parameters used in this benchmark.

1.3 Configuration Diagram

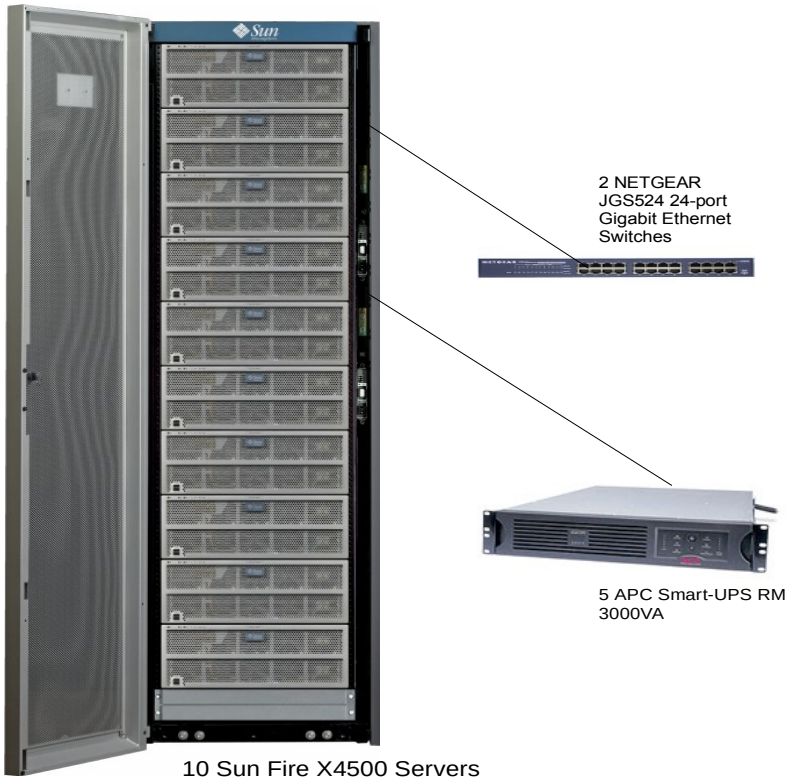
Provide diagrams of both the measured and priced configurations, accompanied by a description of the differences.

10 Sun Fire™ X4500 Servers, each server configured with:

- 2 Dual Core AMD Opteron(tm) Processor 285 processors
- 16 GB memory
- 1 Ethernet controllers
- 48 500GB, 7200 rpm SATA disk drives
- priced configuration
    - 5 APC Smart-UPS RM 3000VA Uninterrupted Power Supplies



Measured Configuration



Priced Configuration

2. Clause 1 Logical Database Design

2.1 Database Definition Statements

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

Appendix B contains the programs and scripts that create and analyze the tables and indexes for the TPC-H database.

2.2 Physical Organization

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

No record clustering or index clustering was used. Column ordering was changed for some tables. Refer to the table create statements in Appendix B for further details.

2.3 Horizontal Partitioning

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

Horizontal partitioning was used for all tables except NATION and REGION. Refer to the table/index create statements in Appendix B for more details.

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 Refresh Functions

3.1 Query Language

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

SQL was the query language used to implement all queries.

3.2 Verifying Method for Random Number Generation

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

TPC supplied versions 2.6.0 of DBGEN and QGEN were used for this TPC-H benchmark.

3.3 Generating Values for Substitution Parameters

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 2.6.0 was used to generate the substitution parameters.

3.4 Query Text and Output Data from Qualification Database

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

Appendix C contains the qualification query text and query output.

3.5 Query Substitution Parameters and Seeds Used

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

Appendix D contains the seed and query substitution parameters.

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 the levels defined in Clause 3.4, additional descriptive detail must be provided.

The queries and transactions were run with isolation level 3 (repeatable read).

3.7 Source Code of Refresh Functions

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

The refresh function is part of the driver code included in Appendix E.

4. Clause 3 Database System Properties

4.1 ACID Properties

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

Source code for the ACID test is included in Appendix B.

4.2 Atomicity

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

4.2.1 Completed Transaction

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

1. The total price from the ORDERS table and the extended price from the LINEITEM table were retrieved for a randomly selected order key.
2. The ACID Transaction was performed using the order key from step 1.
3. The ACID Transaction committed.
4. The total price from the ORDERS table and the extended price from the LINEITEM table were retrieved for the same order key. It was verified that the appropriate rows had been changed.

4.2.2 Aborted Transaction

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

1. The total price from the ORDERS table and the extended price from the LINEITEM table were retrieved for a randomly selected order key.
2. The ACID Transaction was performed using the order key from step 1. The transaction was stopped prior to the commit.
3. The ACID Transaction was ROLLED BACK.
4. The total price from the ORDERS table and the extended price from the LINEITEM table were retrieved for the same order key. It was verified that the appropriate rows had not been changed.

4.3 Consistency

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

4.3.1 Consistency Test

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

1. The consistency of the ORDERS and LINEITEM tables was verified based on a sample of order keys.
2. 100 ACID Transactions were submitted by each of ten execution streams.
3. The consistency of the ORDERS and LINEITEM tables was re-verified.

4.4 Isolation

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

4.4.1 Read-Write Conflict with Commit

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

1. An ACID Transaction was started for a randomly selected O_KEY, L_KEY, and DELTA. The ACID Transaction was suspended prior to COMMIT.
2. An ACID Query was started for the same O_KEY used in step 1. The ACID Query blocked and did not see the uncommitted changes made by the ACID Transaction.
3. The ACID Transaction was resumed and COMMITTED.
4. The ACID Query completed. It returned the data as committed by the ACID Transaction.

4.4.2 Read-Write Conflict with Rollback

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

1. An ACID Transaction was started for a randomly selected O_KEY, L_KEY, and DELTA. The ACID Transaction was suspended prior to ROLLBACK.
2. An ACID Query was started for the same O_KEY used in step 1. The ACID Query did not see the uncommitted changes made by the ACID Transaction.
3. The ACID Transaction was ROLLED BACK.
4. The ACID Query completed.

4.4.3 Write-Write Conflict with Commit

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

1. An ACID Transaction, T1, was started for a randomly selected O_KEY, L_KEY, and DELTA. T1 was suspended prior to COMMIT.
2. Another ACID Transaction, T2, was started using the same O_KEY and L_KEY and a randomly selected DELTA.
3. T2 waited.

4. T1 was allowed to COMMIT and T2 completed.
5. It was verified that T2.L_EXTENDEDPRICE = T1.L_EXTENDEDPRICE + (DELTA1*(T1.L_EXTENDEDPRICE/T1.L_QUANTITY))

4.4.4 Write-Write Conflict with Rollback

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

1. An ACID Transaction, T1, was started for a randomly selected O_KEY, L_KEY, and DELTA. T1 was suspended prior to ROLLBACK.
2. Another ACID Transaction, T2, was started using the same O_KEY and L_KEY and a randomly selected DELTA.
3. T2 waited.
4. T1 was allowed to ROLLBACK and T2 completed.
5. It was verified that T2.L_EXTENDEDPRICE = T1.L_EXTENDEDPRICE.

4.4.5 Concurrent Progress of Read and Write Transactions

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

1. An ACID Transaction, T1, was started for a randomly selected O_KEY, L_KEY, and DELTA. T1 was suspended prior to ROLLBACK.
2. Another Transaction, T2, was started which did the following:

   For random values of PS_PARTKEY and PS_SUPPKEY, all columns of the PARTSUPP table for which PS_PARTKEY and PS_SUPPKEY are equal, are returned.
3. T2 completed.
4. T1 was allowed to COMMIT.
5. It was verified that appropriate rows in ORDERS, LINEITEM and HISTORY tables were changed.

4.4.6 Read-Only Query Conflict with Update Transaction

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

1. A Transaction, T1, executing Q1 against the qualification database, was started using a randomly selected DELTA.
2. An ACID Transaction T2, was started for a randomly selected O_KEY, L_KEY and DELTA.
3. T2 completed and appropriate rows in the ORDERS, LINEITEM and HISTORY tables had been changed.
4. Transaction T1 completed executing Q1.

4.5 Durability

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

4.5.1 Failure of a Durable Medium

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

The disks containing TPC-H tables and log files were mirrored. During the durability test the disk containing one side of a data file mirror was removed from its cabinet. Similarly the disk containing one side of a log file mirror was removed from its cabinet. The test continued uninterrupted, using the remaining side of the mirror.

4.5.2 System Crash

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

The system crash and memory failure tests were combined. Power to the server was turned off by flipping breakers at the main electrical panel during the durability test. When power was restored, the system rebooted and the database was restarted. The durability success file and the HISTORY table were compared successfully

4.5.3 Memory Failure

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

See section 4.5.2

5. Clause 4 Scaling and Database Population

5.1 Ending Cardinality of Tables

The cardinality (i.e., 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	Rows
Orders	4,500,000,000
Lineitem	18,000,048,306
Customer	450,000,000
Part	600,000,000
Supplier	30,000,000
Partsupp	2,400,000,000
Nation	25
Region	5

5.2 Distribution of Tables and Logs Across Media

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

- With the exception of the Nation and Region tables, all tables, indexes and the temporary tablespace were mirrored and striped across 44 disks on each node.
- The Nation and Region tables and indexes resided on a 2 disk mirror on one node
- The system catalog resided on a 2 disk mirror on one node and was NFS mounted by all other nodes.
- The log files resided on a 2 disk mirror on each node.
- For more details refer to disk configuration section in Appendix B.

5.3 Database partition/replication mapping

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

The database was not replicated.

Horizontal partitioning was used for base tables LINEITEM, ORDERS, PARTSUPP, PART, SUPPLIER and CUSTOMER. The details for this partitioning can be understood by examining the syntax of the table and index definition statments in Appendix B.

5.4 RAID Feature

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.

Table/Index	RAID type
tables	RAID 10
indexes	RAID 10
temp tablespace	RAID 10
log	RAID 1
System catalog	RAID 1

5.5 Modifications to the DBGEN

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 supplied DBGEN version 2.6.0 was used to generate the database population for this benchmark.

5.6 Database Load Time

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

The database load time was 3 hours 39 minutes

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 data storage ratio is computed from the following information:

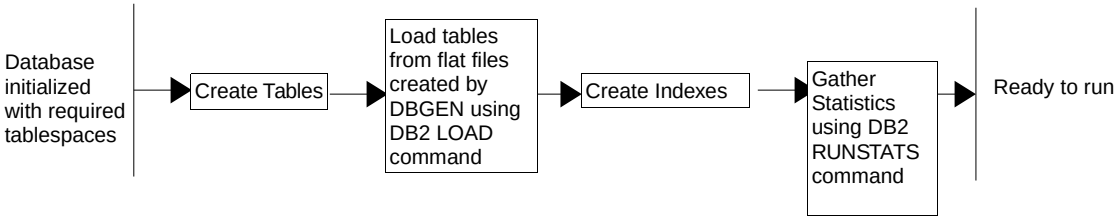
** Disk manufacturer definition of one GB is 10^9 bytes*
***In this calculation one GB is defined as 2^30 bytes*

Disk Type	# Of Disks	Space Per Disk*	Sub-Total Disk Space**
SATA	480	465.76	218.3TB
		Total Space	**218.3 TB**
		Data Storage Ratio	**74.5**

5.8 Database Load Mechanism Details and Illustration

The details of the database load must be described, including a block diagram illustrating the overall process.

The database was loaded using data generation stored on flat files all on the tested and priced configurations.



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 to create and load the data with adjustments for the size difference.

5.10 Dataset verification

Verify that the rows in the loaded database after the performance test are correct by comparing some small number of rows extracted at random from any two files of the corresponding Base, Insert and Delete reference data set files for each table and the corresponding rows of the database.

Verified according to the specification.

5.11 Referential Integrity

Verify referential integrity in the database after the initial load.

Verified according to the specification.

6. Clause 5 Performance Metrics and Execution Rules

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.

1. Auditor requested queries were run against the database to verify the correctness of the load

6.2 Steps in the Power Test

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

The following steps were used to implement the power test:

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

6.3 Timing Intervals for Each Query and Refresh Functions

The timing intervals for each query and for both refresh functions must be reported for the power test.

The power test timing intervals are:

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8a	Q9	Q10	Q11	Q12
Stream 00	322.1	69.5	102.5	34.8	601.9	26.3	677.8	162.0	1794.6	285.2	54.2	52.0
	Q13	**Q14**	**Q15**	**Q16**	**Q17**	**Q18**	**Q19**	**Q20**	**Q21**	**Q22**	**RF1**	**RF2**
Stream 00	758.5	95.5	176.6	317.6	261.7	869.8	262.2	86.6	954.3	291.8	290.8	167.2

6.4 Number of Streams for the Throughput Test

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

Eight 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 stream must be reported for the throughput test.

The throughput test start time and finish time for each stream are contained in the Numerical Quantity Summary earlier in this document.

6.6 Total Elapsed Time of the Measurement Interval

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

The total elapsed time of the throughput test is contained in the Numerical Quantity Summary earlier in this document.

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

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

The start and finish times for each refresh function in the refresh stream are contained in the Numerical Quantity Summary earlier in this document.

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

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

The timing intervals for each query and each refresh function for the throughput test are contained in the Numerical Quantity Summary earlier in this document.

6.9 Performance Metrics

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

The performance metrics, and the numbers on which they are based, are contained in the Numerical Quantity Summary earlier in this document.

6.10 The Performance Metric and Numerical Quantities from Both Runs

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

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

Run ID	QppH@3000GB	QthH@3000GB	QphH@3000GB
Run 1	51,320.0	29,141.8	38,672.4
Run 2	52,562.2	28,813.1	38,916.3

6.11 System Activity Between Performance Tests

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

There was no activity on the SUT between run1 and run2.

7. Clause 6 SUT and Driver Implementation

7.1 Driver

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

Appendix E, "Implementation-Specific Layer/Driver 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 ofTPC-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 t 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

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

Profile-directed optimization was not used.

8. Clause 7 Pricing

8.1 Hardware and Software Used

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

Refer to the Executive Summary.

8.2 Total Three Year Price

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

The total 3-year price of the configuration is $1,136,536. For details of pricing, see the second page of the Executive Summary.

The following generally available discounts to any buyer with like conditions were applied to the priced configuration:

- a 35% Sun support volume and yearly pre-payment discount
- an 18% discount from list for Sun supplied system components

8.3 Availability Date

The committed delivery date for general availability of products used in the price calculations must be reported. When the priced system includes products with different availability dates, the reported availability date for the priced system must be the date at which all components are committed to be available.

All Hardware and Software components are available immediately.

9. Auditor's Information and Attestation Letter

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

The auditor's attestation letter is included at the front of this report.

Appendix A. Solaris 10 and DB2 Parameters

This Appendix contains Solaris kernel parameters and environment variables and DB2 configuration parameters.

==========

DB2 Parameters

(altered from default)

==========

dbmcfg_for_load.sql

```
update database manager configuration using svcename
50000 numdb 1 cpuspeed -1 COMM_BANDWIDTH -1 sheapthres
1250000 instance_memory 0 max_querydegree -1
health_mon off FCM_NUM_BUFFERS 16384 FCM_NUM_CHANNELS
AUTOMATIC num_initagents 0 num_poolagents 0
intra_parallel no diagpath /tpch/db2dump;
```

dbcfg_for_load.sql

```
update database manager configuration using svcename
50000 numdb 1 cpuspeed -1 COMM_BANDWIDTH -1 sheapthres
1250000 instance_memory 0 max_querydegree -1
health_mon off FCM_NUM_BUFFERS 16384 FCM_NUM_CHANNELS
AUTOMATIC num_initagents 0 num_poolagents 0
intra_parallel no diagpath /tpch/db2dump;
```

dbmcg_for_run.sql

```
update database manager configuration using numdb 1
cpuspeed 1.889377e-07 COMM_BANDWIDTH 1.000000e-01
instance_memory 0 max_querydegree any health_mon off
FCM_NUM_BUFFERS 30000 FCM_NUM_CHANNELS AUTOMATIC
num_initagents 0 num_poolagents 0 intra_parallel no;
```

dbcfg_for_run.sql

```
update database configuration for tpcd using
stat_heap_sz 2048 util_heap_sz 5000 database_memory
computed  sortheap 18432 CHNGPGS_THRESH 60
NUM_IOCLEANERS 2 num_ioservers 4 locklist 40000;
```

run_db2set.sh

```
db2set DB2_EXTENDED_OPTIMIZATION=Y
db2set DB2_ANTIJOIN=ON
db2set DB2_STRIPED_CONTAINERS=ON
db2set DB2_LIKE_VARCHAR=Y,Y
db2set DB2BPVARS=bpvars.txt
db2set DB2_FORCE_FCM_BP=ON
db2set DB2COMM=tcpip
db2set DB2_PARALLEL_IO=*
```

bpvars.txt

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

db2nodes.cfg

```
0 45k11ge1 0
1 45k11ge1 1
2 45k11ge1 2
3 45k11ge1 3
4 45k1ge1 0
5 45k1ge1 1
6 45k1ge1 2
7 45k1ge1 3
8 45k2ge1 0
9 45k2ge1 1
10 45k2ge1 2
11 45k2ge1 3
12 45k3ge1 0
13 45k3ge1 1
14 45k3ge1 2
15 45k3ge1 3
16 45k4ge1 0
17 45k4ge1 1
18 45k4ge1 2
19 45k4ge1 3
20 45k5ge1 0
21 45k5ge1 1
22 45k5ge1 2
23 45k5ge1 3
24 45k6ge1 0
25 45k6ge1 1
26 45k6ge1 2
27 45k6ge1 3
28 45k7ge1 0
29 45k7ge1 1
30 45k7ge1 2
31 45k7ge1 3
32 45k8ge1 0
33 45k8ge1 1
34 45k8ge1 2
35 45k8ge1 3
36 45k9ge1 0
37 45k9ge1 1
38 45k9ge1 2
39 45k9ge1 3
```

==========

Solaris Parameters

(altered from default)

/etc/system

```
set md_mirror:md_resync_bufsz = 2048
set md:mirrored_root_flag=1
set sata:sata_func_enable = 0x5
set maxphys=1048576
set md:md_maxphys=1048576
```

/etc/project:

```
:0::::
user.root:1::::
noproject:2::::
default:3::::process.max-sem-
nsems=(priv,8192,deny);project.max-sem-
ids=(priv,8192,deny);project.max-shm-
memory=(priv,12884901888,deny);project.max-shm-
ids=(priv,8192,deny);project.max-locked-
memory=(priv,12884901888,deny);project.max-msg-
ids=(priv,8192,deny)
group.staff:10::::
```

Appendix B. Programs and Scripts

==========

Database Load Scripts

==========
==========

buildtpcd

```
#!/usr/bin/perl
#########################################################
######################
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use,
duplication or
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
#########################################################
######################
# 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_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"};
$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}${delim}$loadsetScri
pt");
        }
        else{
            $rc=system(" rah \" cd ${ddlpath} &
$loadsetScript\" ");
        }
    }
    else{
        $rc=system("${ddlpath}${delim}$loadsetScript")
;
    }
    ($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}$astFile",$onc
e);
    &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";
        if ( $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" ){

&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}firs
t10rows.sql",$once);
}


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

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

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

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

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



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

        if ($logDirScript ne "NULL"){
                system
("$ddlpath${delim}$logDirScript");
                 # using shell script insttead
                #system ("perl
```

```
$ddlpath${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 pagesize 32
k 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}$nodegroupdef",$on
ce);
        }
}

#-----------------------------------------------------
----------------------#
# 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.DD
L";
        }
        $rc = &dodb_conn($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}$buffpooldef",$onc
e);
        }
}

#-----------------------------------------------------
----------------------#
# 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 ",
                        $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",$on
ce);
        }
        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",$a
ll_ln);
        }
#       ($rc == 0) || return $rc;
#       if($loadDBMconfig ne "") || ($loadDBMconfig ne
"NULL"){
#               $rc =
&dodb2file_noconn("$ddlpath${delim}$loadDBMconfig",$on
ce);
#       }
#       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{

#system("$ddlpath${delim}$loadscript");
                        $rc =
&dodb2file_noconn("$ddlpath${delim}$loadscript");
                        ($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${delim}$load
script");
                         $rc =
&dodb2file_noconn("$ddlpath${delim}$loadscript");
                         ($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}$r
unstatShort",$once);
            }
        }
        # run the full runstats, or the remainder of
what wasn't put into the short
        # runstats file. You should be sure that this
runstats will take longer
        # than the short runstats that is running in
the background, otherwise
        # setting the config will happen before this
is done.
        &dodb2file($dbname,"$ddlpath${delim}$runstats"
,$once);
        &outtime("*** Runstats completed");
}

#-----------------------------------------------------
----------------------#
# Function: setConfiguration
#
#-----------------------------------------------------
----------------------#
sub setConfiguration{
        my $ret = 0;
&dodb_noconn("db2 update database 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 == 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 == 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);
        }
        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;
```

tpcd.setup

```
==================================================
TPCD_PLATFORM=sun
TPCD_VERSION=2
TPCD_DBNAME=TPCDQUAL
TPCD_WORKLOAD=H
TPCD_AUDIT_DIR=/export/home/db2inst1/tpch
TPCD_PRODUCT=v5
TPCD_MODE=mln
TPCD_PHYS_NODE=1
TPCD_LN_PER_PN=4
TPCD_SF=1
TPCD_BUILD_STAGE=ALL
TPCD_DBPATH=/tpch/dbpath
TPCD_DDLPATH=/export/home/db2inst1/tpch/custom_qual
TPCD_BUFFERPOOL_DEF=create_bufferpools.ddl
TPCD_NODEGROUP_DEF=create_dbpartgrps.ddl
TPCD_EXPLAIN_DDL=NULL
TPCD_TBSP_DDL=create_tablespaces.ddl
TPCD_DDL=create_tables.ddl
TPCD_QUAL_TBSP_DDL=create_tablespaces.ddl
TPCD_QUAL_DDL=create_tables.ddl
TPCD_INDEXDDL=create_indexes.ddl
TPCD_EXTRAINDEX=no
TPCD_ADD_RI=NULL
TPCD_AST=NULL
TPCD_RUNSTATS=runstats.ddl
TPCD_RUNSTATSHORT=NULL
TPCD_DBGEN=/export/home/db2inst1/tpch/appendix.v2.6
TPCD_INPUT=/tpch/flat
TPCD_QUAL_INPUT=NULL
TPCD_TAILOR_DIR=/export/home/db2inst1/tpch/tailor
TPCD_EARLYINDEX=no
TPCD_LOAD_DB2SET_SCRIPT=run_db2set.sh
TPCD_LOAD_CONFIGFILE=dbcfg_for_load.sql
```

```
TPCD_LOAD_DBM_CONFIGFILE=dbmcfg_for_load.sql
TPCD_LOAD_QUALCONFIGFILE=dbcfg_for_load.sql
TPCD_LOAD_DBM_QUALCONFIGFILE=dbmcfg_for_load.sql
TPCD_LOADSTATS=NO
TPCD_TEMP=/export/home/db2inst1/tpch/tmp
TPCD_SORTBUF=4096
TPCD_LOAD_PARALLELISM=0
TPCD_COPY_DIR=/dev/null
TPCD_FASTPARSE=yes
TPCD_LOG_DIR_SETUP_SCRIPT=set_newlogpath.sh
TPCD_BACKUP_DIR=NULL
TPCD_LOGPRIMARY=NULL
TPCD_LOGFILSIZ=NULL
TPCD_LOGSECOND=NULL
TPCD_LOG_DIR=NULL
TPCD_LOG_QUAL_DIR=NULL
TPCD_LOG=no
TPCD_DB2SET_SCRIPT=NULL
TPCD_CONFIGFILE=dbcfg_for_run.sql
TPCD_DBM_CONFIG=dbmcfg_for_run.sql
TPCD_QUALCONFIGFILE=dbcfg_for_run.sql
TPCD_DBM_QUALCONFIG=dbcfg_for_run.sql
TPCD_MACHINE=NULL
TPCD_SMPDEGREE=1
TPCD_AGENTPRI=NULL
TPCD_ACTIVATE=no
TPCD_AUDIT=no
TPCD_TMP_DIR=/export/home/db2inst1/tpch/tmp
TPCD_SHARED_TEMP_FULL_PATHNAME=/export/home/db2inst1/s
qllib/tmp
TPCD_QUERY_TEMPLATE_DIR=standard.V2
TPCD_QUAL_DBNAME=TPCDQUAL
TPCD_NUMSTREAM=8
TPCD_FLATFILES=/ff1/data/updates
TPCD_STAGING_TABLE_DDL=createUFtables
TPCD_PRELOAD_STAGING_TABLE_SCRIPT=NULL
TPCD_DELETE_STAGING_TABLE_SQL=NULL
TPCD_UPDATE_IMPORT=false
TPCD_SPLIT_UPDATES=40
TPCD_CONCURRENT_INSERTS=20
TPCD_CONCURRENT_INSERTS_LOAD=4
TPCD_SPLIT_DELETES=40
TPCD_CONCURRENT_DELETES=20
TPCD_GEN_UPDATEPAIRS=22
TPCD_GENERATE_SEED_FILE=yes
TPCD_RUN_ON_MULTIPLE_NODES=NO
TPCD_STATS_INTERVAL=31
TPCD_STATS_THRU_INT=300
TPCD_GATHER_STATS=off
TPCD_UFTEMP=DATA_INDEX
TPCD_HAVECOMPILER=yes
TPCD_SLEEP=5
TPCD_INLISTMAX=default
TPCD_LOAD_SCRIPT=load_tables.clp
TPCD_LOAD_SCRIPT_QUAL=load_tables.clp
TPCD_ROOTPRIV=no
TPCD_DB2LOG=/tpch/db2dump
TPCD_APPEND_ON=no
```

set_newlogpath.sh

```
db2_all "<<+0<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d801"
db2_all "<<+1<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d802"
db2_all "<<+2<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d803"
db2_all "<<+3<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d804"
db2_all "<<+4<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d801"
db2_all "<<+5<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d802"
db2_all "<<+6<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d803"
db2_all "<<+7<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d804"
db2_all "<<+8<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d801"
db2_all "<<+9<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d802"
db2_all "<<+10<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d803"
db2_all "<<+11<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d804"
db2_all "<<+12<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d801"
db2_all "<<+13<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d802"
db2_all "<<+14<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d803"
db2_all "<<+15<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d804"
db2_all "<<+16<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d801"
db2_all "<<+17<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d802"
db2_all "<<+18<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d803"
db2_all "<<+19<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d804"
db2_all "<<+20<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d801"
db2_all "<<+21<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d802"
db2_all "<<+22<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d803"
db2_all "<<+23<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d804"
db2_all "<<+24<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d801"
db2_all "<<+25<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d802"
db2_all "<<+26<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d803"
db2_all "<<+27<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d804"
db2_all "<<+28<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d801"
db2_all "<<+29<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d802"
db2_all "<<+30<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d803"
db2_all "<<+31<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d804"
db2_all "<<+32<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d801"
db2_all "<<+33<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d802"
db2_all "<<+34<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d803"
db2_all "<<+35<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d804"
db2_all "<<+36<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d801"
db2_all "<<+37<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d802"
db2_all "<<+38<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d803"
db2_all "<<+39<db2 update db cfg for tpcd using
newlogpath /dev/md/rdsk/d804"
```

==========

create_dbpartgrps.ddl

```
connect to tpcd;
-- create database partition groups
CREATE DATABASE PARTITION GROUP "NG_ALL" ON
DBPARTITIONNUMS (0 to 39);
CREATE DATABASE PARTITION GROUP "NG_NODE1" ON
DBPARTITIONNUMS (1);
COMMIT;
```

==========

create_bufferpools.ddl

```
connect to tpcd;
alter bufferpool ibmdefaultbp size 200;
commit work;
create bufferpool  BP32K all nodes size 43900 pagesize
32K;
commit work;
ALTER BUFFERPOOL BP32K NUMBLOCKPAGES 8000 BLOCKSIZE
16;
COMMIT WORK;
```

==========

create_tablespaces.ddl

```
================================================
connect to tpcd;

CREATE TABLESPACE DATA_INDEX IN DATABASE PARTITION
GROUP NG_ALL PAGESIZE 32768 MANAGED BY DATABASE
        USING (DEVICE '/dev/md/rdsk/d100' 3906336) ON
DBPARTITIONNUMS (0,4,8,12,16,20,24,28,32,36)
        USING (DEVICE '/dev/md/rdsk/d200' 3906336) ON
DBPARTITIONNUMS (1,5,9,13,17,21,25,29,33,37)
        USING (DEVICE '/dev/md/rdsk/d300' 3906336) ON
DBPARTITIONNUMS (2,6,10,14,18,22,26,30,34,38)
        USING (DEVICE '/dev/md/rdsk/d400' 3906336) ON
DBPARTITIONNUMS (3,7,11,15,19,23,27,31,35,39)
        EXTENTSIZE 16
        PREFETCHSIZE 32
        BUFFERPOOL BP32K;

CREATE TEMPORARY TABLESPACE TEMP32K PAGESIZE 32768
MANAGED BY DATABASE
        USING (DEVICE '/dev/md/rdsk/d101' 16384000)
ON DBPARTITIONNUMS (0,4,8,12,16,20,24,28,32,36)
        USING (DEVICE '/dev/md/rdsk/d201' 16384000)
ON DBPARTITIONNUMS (1,5,9,13,17,21,25,29,33,37)
        USING (DEVICE '/dev/md/rdsk/d301' 16384000)
ON DBPARTITIONNUMS (2,6,10,14,18,22,26,30,34,38)
        USING (DEVICE '/dev/md/rdsk/d401' 16384000)
ON DBPARTITIONNUMS (3,7,11,15,19,23,27,31,35,39)
        EXTENTSIZE 16
        PREFETCHSIZE 32
        BUFFERPOOL BP32K;

CREATE TABLESPACE SMALL_DATA
IN NODEGROUP NG_NODE1
MANAGED BY SYSTEM USING
('/tpch/dbpath/small_data')
;

drop tablespace tempspace1;

commit work;

================================================
```

create_tables.ddl

```
    connect to tpcd;

    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 DATA_INDEX
    INDEX IN DATA_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 DATA_INDEX
    INDEX IN DATA_INDEX
    PARTITIONING KEY(S_SUPPKEY);

    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 DATA_INDEX
    INDEX IN DATA_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 DATA_INDEX
    INDEX IN DATA_INDEX
    PARTITIONING KEY(C_CUSTKEY);

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

    CREATE TABLE TPCD.LINEITEM ( L_ORDERKEY    BIGINT
NOT NULL,
    L_PARTKEY     INTEGER NOT NULL,
    L_SUPPKEY     INTEGER NOT NULL,
    L_LINENUMBER  INTEGER NOT NULL,
    L_QUANTITY    FLOAT NOT NULL,
    L_EXTENDEDPRICE  FLOAT NOT NULL,
    L_DISCOUNT    FLOAT NOT NULL,
    L_TAX         FLOAT NOT NULL,
    L_RETURNFLAG  CHAR(1) NOT NULL,
    L_LINESTATUS  CHAR(1) NOT NULL,
    L_SHIPDATE    DATE NOT NULL,
    L_COMMITDATE  DATE NOT NULL,
    L_RECEIPTDATE DATE NOT NULL,
    L_SHIPINSTRUCT CHAR(25) NOT NULL,
    L_SHIPMODE     CHAR(10) NOT NULL,
    L_COMMENT      VARCHAR(44) NOT NULL WITH DEFAULT
    --                               PRIMARY KEY
(L_ORDERKEY, L_LINENUMBER)
    )
    IN DATA_INDEX
    INDEX IN DATA_INDEX
    PARTITIONING KEY(L_ORDERKEY) USING HASHING
    ORGANIZE BY (  (L_SHIPDATE) );

   COMMIT WORK;
```

load_tables.clp

```
connect to tpcd;

values(current timestamp, 'TS*** Load Supplier
Started');
LOAD FROM
supplier.tbl.sorted
OF DEL
MODIFIED BY COLDEL|
FASTPARSE ANYORDER
MESSAGES
/export/home/db2inst1/tpch/temp/load_supplier.msg
REPLACE INTO tpcd.SUPPLIER
NONRECOVERABLE DATA BUFFER 256
```

```
PARTITIONED DB CONFIG MODE LOAD_ONLY
PART_FILE_LOCATION /tpch/flat;
values(current timestamp, 'TS*** Load Supplier
Finished');



values(current timestamp, 'TS*** Load Lineitem
Started');
LOAD FROM
lineitem.tbl.sorted
OF DEL
MODIFIED BY COLDEL|
FASTPARSE ANYORDER
MESSAGES
/export/home/db2inst1/tpch/temp/load_lineitem.msg
REPLACE INTO tpcd.LINEITEM
NONRECOVERABLE DATA BUFFER 256
PARTITIONED DB CONFIG MODE LOAD_ONLY
PART_FILE_LOCATION /tpch/flat;
values(current timestamp, 'TS*** Load Lineitem
Finished');


values(current timestamp, 'TS*** Load Orders
Started');
LOAD FROM
orders.tbl.sorted
OF DEL
MODIFIED BY COLDEL|
FASTPARSE ANYORDER
MESSAGES
/export/home/db2inst1/tpch/temp/load_orders.msg
REPLACE INTO tpcd.ORDERS
NONRECOVERABLE DATA BUFFER 256
PARTITIONED DB CONFIG MODE LOAD_ONLY
PART_FILE_LOCATION /tpch/flat;
values(current timestamp, 'TS*** Load Orders
Finished');


values(current timestamp, 'TS*** Load Customer
Started');
LOAD FROM
customer.tbl.sorted
OF DEL
MODIFIED BY COLDEL|
FASTPARSE ANYORDER
MESSAGES
/export/home/db2inst1/tpch/temp/load_customer.msg
REPLACE INTO tpcd.CUSTOMER
NONRECOVERABLE DATA BUFFER 256
PARTITIONED DB CONFIG MODE LOAD_ONLY
PART_FILE_LOCATION /tpch/flat;
values(current timestamp, 'TS*** Load Customer
Finished');


values(current timestamp, 'TS*** Load Partsupp
Started');
LOAD FROM
partsupp.tbl.1
OF DEL
MODIFIED BY COLDEL|
FASTPARSE ANYORDER
MESSAGES
/export/home/db2inst1/tpch/temp/load_partsupp.msg
REPLACE INTO tpcd.PARTSUPP
NONRECOVERABLE DATA BUFFER 256
PARTITIONED DB CONFIG MODE LOAD_ONLY
PART_FILE_LOCATION /tpch/flat;
values(current timestamp, 'TS*** Load Partsupp
Finished');


values(current timestamp, 'TS*** Load Part Started');
LOAD FROM
part.tbl.sorted
OF DEL
MODIFIED BY COLDEL|
FASTPARSE ANYORDER
MESSAGES /export/home/db2inst1/tpch/temp/load_part.msg
REPLACE INTO tpcd.PART
NONRECOVERABLE DATA BUFFER 256
PARTITIONED DB CONFIG MODE LOAD_ONLY
PART_FILE_LOCATION /tpch/flat;
values(current timestamp, 'TS*** Load Part Finished');


values(current timestamp, 'TS*** Load Region
Started');
LOAD FROM
/tpch/flat/region.tbl
OF DEL
MODIFIED BY COLDEL|
FASTPARSE ANYORDER
MESSAGES
/export/home/db2inst1/tpch/temp/load_region.msg
REPLACE INTO tpcd.REGION
NONRECOVERABLE DATA BUFFER 256;
values(current timestamp, 'TS*** Load Region
Finished');


values(current timestamp, 'TS*** Load Nation
Started');
LOAD FROM
/tpch/flat/nation.tbl
OF DEL
MODIFIED BY COLDEL|
FASTPARSE ANYORDER
MESSAGES
/export/home/db2inst1/tpch/temp/load_nation.msg
REPLACE INTO tpcd.NATION
NONRECOVERABLE DATA BUFFER 256;
values(current timestamp, 'TS*** Load Nation
Finished');

commit work;

connect reset;

terminate;
```

==========

create_indexes.ddl

```
connect to tpcd;

values(current timestamp);

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

alter table tpcd.region add primary key (r_regionkey);
commit work;

values(current timestamp);

CREATE UNIQUE INDEX TPCD.N_NK ON TPCD.NATION
(N_NATIONKEY ASC) PCTFREE 0;
commit work;

alter table tpcd.nation add primary key (n_nationkey);
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;

alter table tpcd.supplier add primary key (s_suppkey);
commit work;

values(current timestamp);

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

values(current timestamp);

CREATE UNIQUE INDEX TPCD.PS_PKSK ON TPCD.PARTSUPP
```

```
(PS_PARTKEY ASC, PS_SUPPKEY ASC) PCTFREE 0;
commit work;

values(current timestamp);

alter table tpcd.partsupp add primary key (ps_partkey,
ps_suppkey);
commit work;

values(current timestamp);

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

values(current timestamp);

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

values(current timestamp);

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

values(current timestamp);

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

values(current timestamp);

alter table tpcd.part add primary key (p_partkey);
commit work;

values(current timestamp);

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

values(current timestamp);

alter table tpcd.customer add primary key (c_custkey);
commit work;

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);
commit work;

values(current timestamp);

alter table tpcd.orders add primary key (o_orderkey);
commit work;

CREATE INDEX TPCD.O_CK ON TPCD.ORDERS (O_CUSTKEY ASC);
commit work;

values(current timestamp);

CREATE UNIQUE INDEX TPCD.L_OKLN ON TPCD.LINEITEM
(L_ORDERKEY ASC, L_LINENUMBER ASC);
commit work;

values(current timestamp);

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

values(current timestamp);

select
substr(tbname,1,10),substr(name,1,18),create_time from
sysibm.sysindexes where tbcreator='TPCD' order by 3;
```

runstats.ddl

```
connect to tpcd;

values (current timestamp, 'TS*** runstats nation
START like ');
RUNSTATS ON TABLE TPCD.NATION WITH DISTRIBUTION on all
columns
and columns (
n_name like statistics,
n_comment like statistics )
AND detailed INDEXES ALL;
commit;
values (current timestamp, 'TS*** runstats done nation
');
RUNSTATS ON TABLE TPCD.REGION WITH DISTRIBUTION on all
columns
and columns (
r_name like statistics,
r_comment like statistics )
AND detailed INDEXES ALL;
commit;
RUNSTATS ON TABLE TPCD.SUPPLIER WITH DISTRIBUTION on
all
columns
and columns (
s_name like statistics,
s_address like statistics,
s_phone like statistics,
s_comment like statistics)
AND detailed INDEXES ALL;
commit;
values (current timestamp, 'TS*** runstats done part
');
RUNSTATS ON TABLE TPCD.PART WITH DISTRIBUTION on all
columns
and columns (
p_name like statistics,
p_mfgr like statistics,
p_brand like statistics,
p_type like statistics,
p_container like statistics,
p_comment like statistics)
AND detailed INDEXES ALL;
commit;
values (current timestamp, 'TS*** runstats done
partsupp ');
RUNSTATS ON TABLE TPCD.PARTSUPP WITH DISTRIBUTION on
all
columns
and columns (
ps_comment like statistics)
AND detailed INDEXES ALL;
commit;
values (current timestamp, 'TS*** runstats done
customer ');
RUNSTATS ON TABLE TPCD.CUSTOMER WITH DISTRIBUTION on
all
columns
and columns (
c_name like statistics,
c_address like statistics,
c_phone like statistics,
c_mktsegment like statistics,
c_comment like statistics)
AND detailed INDEXES ALL;
commit;
values (current timestamp, 'TS*** runstats done orders
');
RUNSTATS ON TABLE TPCD.ORDERS WITH DISTRIBUTION on all
columns
and columns (
o_orderstatus like statistics,
o_orderpriority like statistics,
o_clerk like statistics,
o_comment like statistics)
AND detailed INDEXES ALL;
commit;
values (current timestamp, 'TS*** runstats done
lineitem ');
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 detailed INDEXES ALL;
COMMIT WORK;
values (current timestamp, 'TS*** runstats END like');
```

createuftables

```
connect to tpcd;

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

commit;

CREATE TABLE TPCDTEMP.ORDERS_NEW ( APP_ID INTEGER NOT
NULL,
                        O_ORDERKEY       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)
        IN DATA_INDEX
        INDEX IN DATA_INDEX
        PARTITIONING KEY(O_ORDERKEY) USING HASHING;

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 TABLE TPCDTEMP.ORDERS_DEL ( APP_ID      INTEGER
NOT NULL,
                                   O_ORDERKEY  INTEGER
NOT NULL)
        IN DATA_INDEX
        INDEX IN DATA_INDEX
        PARTITIONING KEY(O_ORDERKEY) USING HASHING;

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



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)
        IN DATA_INDEX
        INDEX IN DATA_INDEX
        PARTITIONING KEY(L_ORDERKEY);

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

COMMIT WORK;

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

COMMIT WORK;

connect reset;
```

preloaduf

```
#!/bin/ksh
#########################################################
######################
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use,
duplication or
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
#########################################################
######################

# 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=/export/home/db2inst1/tpch/tools

${toolsDir}/ploaduf1 20
${toolsDir}/ploaduf2 20

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"
```

==========

ACID Test Source Code

==========

acid.sqc

```
/*
########################################################
######################## */
/* ## Licensed Materials - Property of IBM
*/
/* ##
*/
/* ## Governed under the terms of the International
*/
/* ## License Agreement for Non-Warranted Sample Code.
*/
/* ##
*/
/* ## (C) COPYRIGHT International Business Machines
Corp. 1990 - 2005            */
/* ## All Rights Reserved.
*/
/* ##
*/
/* ## US Government Users Restricted Rights - Use,
duplication or                */
/* ## disclosure restricted by GSA ADP Schedule
Contract with IBM Corp.          */
/*
########################################################
######################## */
/*******************************************************
******************/
/*      File: acid.sqc
*/
/*******************************************************
******************/

/*   changes:
 *
 *  961109  jel    add EXEC SQL CLOSE for each cursor
in acidT
 *                 to avoid bug in db2pe v1r2
 *  980225  gav    port to NT
 *  981103  kal    added ast_acidQ  for isolation test
7
 *  981103  kal    changed ast query to be the same as
that used in
 *                 consistency tests.  Fixed so the
long lEprice is
 *                 cast to a double.  Changed so uses
3 decimal points of
 *                 precision.
 *
 */

#include "acid.h"

#if (defined(SQLPTX) || defined(SQLWINT) ||
defined(SQLSUN) || defined(Linux))
double nearest(double);
#endif /* SQLPTX */


#define DEADLOCK -911

/*
#define TRUNC2(d) ((floor((d)*100.0))/100.0)
*/
/*
#define TRUNC2(d) ((floor(nearest((d)*100.0)))*0.01)
*/
/*
#define TRUNC2(d)
((floor(nearest((d)*1000.0)/10.0)/100.0))
*/
#define TRUNC2(d)
((floor(nearest((d)*100000.0)/1000.0)/100.0))

void sqlerror(char * , struct sqlca *);

EXEC SQL INCLUDE SQLCA;
EXEC SQL BEGIN DECLARE SECTION;
char dbname[8];   /* = "tpcd"; */
EXEC SQL END DECLARE SECTION;

#ifdef SQLWINT

 /*
 ** redefine gettimeofday so I don't have to
 ** change too much aix-specific code
 */
/*#typedef struct timeval { unsigned tv_sec; unsigned
tv_usec; }; */
typedef struct timezone { int dummy; };
struct timeb timer;

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

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


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

   okey = acid->o_key;

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

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

   gettimeofday(&tv, &tz);
   time(&timeT);
   fprintf(out,"\n---------- START of acidQ tag: %d
----------\n\n",mypid);
   fprintf(out, "acidQ tag: %d, begin transaction
time: (%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(DEC
IMAL
         (INTEGER(100*DECIMAL(L_EXTENDEDPRICE,20,3)),
20,3) *
         (1-L_DISCOUNT)) *
(1+L_TAX)),20,3)/100.0),20,3) * 1000)
       into :lEprice
     FROM
       TPCD.LINEITEM
     WHERE
       L_ORDERKEY = :okey;

   if (sqlca.sqlcode != 0) {
      rc = sqlca.sqlcode;
      fprintf(out,"acidQ **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
      sqlerror("acidQ: select sum(l_extendedprice)",
&sqlca);
      goto Qerror;
   }
   eprice = (double)lEprice / 1000.0;  /* translate to
double for printout*/

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

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

   rc = 0;
   goto Qexit;

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

 Qexit:
   fprintf(out,"\n---------- END of acidQ tag: %d
----------\n\n",mypid);
   fflush(out);fclose(out);
   return(rc);
}

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


   EXEC SQL BEGIN DECLARE SECTION;
   double    ast_lEprice;
   double    ast_eprice;
   EXEC SQL END DECLARE SECTION;


   /* mypid = getpid(); */

   mypid = acid->tag;

   sprintf(out_fn,
"%s%cast_acidQ.out.%d",getenv("TPCD_TMP_DIR"),del(),my
pid);
   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[50];
   struct timeval tv;
   struct timezone tz;
   int mypid;
   int rc = 0;

   EXEC SQL BEGIN DECLARE SECTION;
   sqlint32      o_key, l_key, delta;
   sqlint32      l_partkey, l_suppkey;
   double    l_quantity, l_tax, l_discount,
l_extendedprice;
   double    o_totalprice;
   double    new_quantity, rprice, cost, new_extprice,
new_ototal, 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:
      if (acid->logging) {
         gettimeofday(&tv, &tz);
         time(&timeT);
         fprintf(out,"acidT tag: %d, immediately
before COMMIT: (%us %06uu) %s",
                 mypid, tv.tv_sec, tv.tv_usec,
ctime(&timeT));
      }
      EXEC SQL CLOSE L_CURSOR;
      EXEC SQL CLOSE O_CURSOR;
      EXEC SQL COMMIT;
      if (sqlca.sqlcode != 0) {
         if(sqlca.sqlcode == DEADLOCK) goto
retry_tran;
         rc = sqlca.sqlcode;
         if (acid->logging) {
           fprintf(out,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
         } else {
           fprintf(stderr,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
         } /* endif */
         sqlerror("acidT: COMMIT", &sqlca);
         goto Terror;
      }
      if (acid->logging) {
         gettimeofday(&tv, &tz);
         time(&timeT);
         fprintf(out,"acidT tag: %d, after COMMIT:
(%us %06uu) %s",
                 mypid, tv.tv_sec, tv.tv_usec,
ctime(&timeT));
      }
      break;
    case 3:
      if (acid->logging) {
         gettimeofday(&tv, &tz);
         time(&timeT);
         fprintf(out,"acidT tag: %d, wait before
ROLLBACK: (%us %06uu) %s",
                 mypid, tv.tv_sec, tv.tv_usec,
ctime(&timeT));
      }
      sleep(60);
    case 2:
      if (acid->logging) {
         gettimeofday(&tv, &tz);
         time(&timeT);
         fprintf(out,"acidT tag: %d, immediately
before ROLLBACK: (%us %06uu) %s",
                 mypid, tv.tv_sec, tv.tv_usec,
ctime(&timeT));
      }
      EXEC SQL CLOSE L_CURSOR;
      EXEC SQL CLOSE O_CURSOR;
      EXEC SQL rollback work;
      if (sqlca.sqlcode != 0) {
         if(sqlca.sqlcode == DEADLOCK) goto
retry_tran;
         rc = sqlca.sqlcode;
         if (acid->logging) {
           fprintf(out,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
         } else {
           fprintf(stderr,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
         } /* endif */
         sqlerror("acidT: ROLLBACK", &sqlca);
         goto Terror;
      }
      if (acid->logging) {
         gettimeofday(&tv, &tz);
         time(&timeT);
         fprintf(out,"acidT tag: %d, after ROLLBACK:
(%us %06uu) %s",
                 mypid, tv.tv_sec, tv.tv_usec,
ctime(&timeT));
      }
         break;
   }

   acid->l_partkey = l_partkey;
   acid->l_suppkey = l_suppkey;
   acid->l_quantity = l_quantity;
   acid->l_tax = l_tax;
   acid->l_discount = l_discount;
   acid->l_extendedprice = l_extendedprice;
   acid->o_totalprice = o_totalprice;

   rc = 0;
   goto Texit;

 Terror:
      EXEC SQL CLOSE L_CURSOR;
      EXEC SQL CLOSE O_CURSOR;
   EXEC SQL rollback work;
   if (sqlca.sqlcode != 0) sqlerror("acidT: ROLLBACK
FAILED", &sqlca);

 Texit:
   if (acid->logging) {
      fprintf(out,"\n---------- END of acidT tag: %d
----------\n\n",mypid);
      fflush(out);fclose(out);
   }
   return(rc);
}

/*-------------------------------------------------
----------*/
/*       updateQ
*/
/*-------------------------------------------------
----------*/
int updateQ (struct update_struct *us)
{
   FILE *out;
   time_t timeT;
```

```
   struct timeval tv;
   struct timezone tz;
   int qnum;
   int rc = 0;
   int i;
   int secs2sleep;
   char buff[256];
   struct acidtype {int logging;} a, *acid;

   EXEC SQL BEGIN DECLARE SECTION;
   double    acctbal;
   double    discount;
   double    price;
   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#0
00000802');
          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 */
```

acid.h

```
/*
##########################################################
####################### */
/* ## Licensed Materials - Property of IBM
*/
/* ##
*/
/* ## Governed under the terms of the International
*/
/* ## License Agreement for Non-Warranted Sample Code.
*/
/* ##
*/
/* ## (C) COPYRIGHT International Business Machines
Corp. 1990 - 2005            */
/* ## All Rights Reserved.
*/
/* ##
*/
/* ## US Government Users Restricted Rights - Use,
duplication or                 */
/* ## disclosure restricted by GSA ADP Schedule
Contract with IBM Corp.          */
/*
##########################################################
####################### */
/*****************************************************
******************/
/*      File: acid.h
*/
/*****************************************************
******************/

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

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

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

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

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

struct update_struct {
   int     qnum;
};

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

```
};

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

/*
** in acid.sqc
*/

int updateQ (struct update_struct *us);

char del(void);

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

gen6rnd.c

```
/*
############################################################
######################### */
/* ## Licensed Materials - Property of IBM
*/
/* ##
*/
/* ## Governed under the terms of the International
*/
/* ## License Agreement for Non-Warranted Sample Code.
*/
/* ##
*/
/* ## (C) COPYRIGHT International Business Machines
Corp. 1990 - 2005             */
/* ## All Rights Reserved.
*/
/* ##
*/
/* ## US Government Users Restricted Rights - Use,
duplication or                */
/* ## disclosure restricted by GSA ADP Schedule
Contract with IBM Corp.           */
/*
############################################################
######################### */
#ifndef WINNT
#include <unistd.h>
#endif
#include <stdio.h>

int rand_integer (int val_lo, int val_hi);

int main(int argc, char *argv[])

{
   unsigned int lo, hi, n, r, i;
   unsigned int unique[6];
   unsigned int cnt, new_number;

   n = 6;

   if (argc != 3)
   {
     fprintf(stderr,"usage: gen6rnd start end \n");
     goto exit;
   }

   srand(getpid());

   lo = atoi(argv[1]);
   hi = atoi(argv[2]);

   if ( (hi-lo+1) <  n )
   {
     fprintf(stderr,"cannot generate %d unique numbers
from %d to %d \n",
             n,lo,hi );
     goto exit;
   }

   #ifdef DEBUG
   fprintf(stderr,"Generating %d random numbers
between %d and %d\n",
           n, lo, hi);
   #endif

   unique[0] = rand_integer(lo, hi);
   cnt = 1;
   while ( cnt <= n )
   {
      new_number = 1;
      r = rand_integer(lo, hi);
      for ( i=0; i < cnt; i ++ )
      {
        if ( unique[i] == r )
        {
          new_number = 0;
          break;
        }
      }

      if ( new_number )
      {
        unique[cnt] = r;
        cnt ++ ;
        printf ("%04d\n", r);
      }
   }
exit:
  exit(0);  /* return traps on nt */
  return 0;
}

int rand_integer (int val_lo, int val_hi)
{
   return( (rand() % (val_hi-val_lo+1)) + val_lo );
}
```

gendata.c

```
static char sqla_program_id[162] =
{
 42,0,65,68,65,76,65,73,71,69,78,68,65,84,65,32,110,65
,67,72,
 80,99,74,88,48,49,49,49,49,32,50,32,8,0,68,66,50,73,7
8,83,
 84,49,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
 0,0
};

#include "sqladef.h"

static struct sqla_runtime_info sqla_rtinfo =
{{'S','Q','L','A','R','T','I','N'}, sizeof(wchar_t),
0, {' ',' ',' ',' ',' '}};


static const short sqlIsLiteral   = SQL_IS_LITERAL;
static const short sqlIsInputHvar = SQL_IS_INPUT_HVAR;

/*
############################################################
######################### */
/* ## Licensed Materials - Property of IBM
*/
/* ##
*/
/* ## Governed under the terms of the International
*/
/* ## License Agreement for Non-Warranted Sample Code.
*/
/* ##
```

```
*/
/* ## (C) COPYRIGHT International Business Machines
Corp. 1990 - 2005             */
/* ## All Rights Reserved.
*/
/* ##
*/
/* ## US Government Users Restricted Rights - Use,
duplication or                */
/* ## disclosure restricted by GSA ADP Schedule
Contract with IBM Corp.           */
/*
########################################################
####################### */
#include <stdio.h>
#include <stdlib.h>
#ifndef SQLWINT
#include <unistd.h>
#endif
#include <string.h>
#include <time.h>


/*
EXEC SQL INCLUDE SQLCA;
*/

/* SQL Communication Area - SQLCA - structures and
constants */
#include "sqlca.h"
struct sqlca sqlca;



/*
EXEC SQL BEGIN DECLARE SECTION;
*/

char dbname[8];
sqlint32 okey, linenum;

/*
EXEC SQL END DECLARE SECTION;
*/



extern int getpid();
static int rand_integer ( int val_lo, int val_hi );
int connectDB(void);
int disconnectDB(void);


main(int argc, char *argv[])

{

   FILE *fnT;
   FILE *fnQ;
/*   long offset;*/
   int numstreams;
   int lo, hi, n, i, j, intrvl;
/*   int r;*/
   char transfile[50];
   char queryfile[20];

   lo = 1;
   hi = 6000000;
   n = 100;
   numstreams = 1;
#ifdef SQLWINT
   srand((int)time(NULL));
#else
   srand(getpid());
   srandom(getpid());
#endif
   if (argc>5) argc=5;
   switch (argc) {
    case 5:
      numstreams = atoi(argv[4]);
    case 4:
      hi  = atoi(argv[3]);
    case 3:
      lo = atoi(argv[2]);
    case 2:
      n = (atoi(argv[1]));
      break;
   }
   intrvl = n / 10;
   sprintf(transfile,"%d.in.%d",n, numstreams);
   fnT=fopen(transfile,"w");
   sprintf(queryfile,"10.%d.in",n);
   fnQ=fopen(queryfile,"w");
/*
   printf("\ngendata running with the following
parameters:\n");
   printf("\tgenerating %d files each
with\n",numstreams);
   printf("\t    %d random orderkeys between %d and
%d\n",n,lo,hi);
   printf("\tand creating %s and %s output files\n\n",
transfile, queryfile);
*/


   /* connect to dbname */
   if (connectDB())
    {
     goto theExit;
    }
   j = 0;
   i=1;
   while (i <= n)
    {
      /* query lineitem table with random orderkey */
      okey=rand_integer(lo,hi);
      linenum=0;

/*
EXEC SQL
        SELECT
           MAX(L_LINENUMBER)
           into :linenum
        FROM
           TPCD.LINEITEM
        WHERE
           L_ORDERKEY = :okey;
*/

{
  sqlastrt(sqla_program_id, &sqla_rtinfo, &sqlca);
  sqlaaloc(2,1,1,0L);
    {
      struct sqla_setdata_list sql_setdlist[1];
      sql_setdlist[0].sqltype = 496;
sql_setdlist[0].sqllen = 4;
      sql_setdlist[0].sqldata = (void*)&okey;
      sql_setdlist[0].sqlind = 0L;
      sqlasetdata(2,0,1,sql_setdlist,0L,0L);
    }
  sqlaaloc(3,1,2,0L);
    {
      struct sqla_setdata_list sql_setdlist[1];
      sql_setdlist[0].sqltype = 496;
sql_setdlist[0].sqllen = 4;
      sql_setdlist[0].sqldata = (void*)&linenum;
      sql_setdlist[0].sqlind = 0L;
      sqlasetdata(3,0,1,sql_setdlist,0L,0L);
    }
  sqlacall((unsigned short)24,1,2,3,0L);
  sqlastop(0L);
}


/*
      printf("orderkey= %d, linenumber= %d\n", okey,
linenum);
*/

      if (sqlca.sqlcode == 0)
       {
        /* returned a row */
        j++;
        /* The LINENUMBER read is the maximum value so
no generate a random
           number between 1 - (maximum LINENUMBER). */
        fprintf(fnT,"%06d %d\n", okey,
rand_integer(1,linenum));

/*
        printf("trans file: orderkey= %06d\n", okey);
*/
```

```
        if (intrvl == j) {
           j = 0;
           fprintf(fnQ,"%06d ", okey);
/*
           printf(" query file: orderkey= %06d\n",
okey);
*/
         }
        i++;
       } /* if return code 0 ... row returned */
      else if (sqlca.sqlcode != -305)
       {
        /* sqlcode=-305 means null value assigned to
host variable */
        /* do nothing */
        /* if not 100, unexpected error */
        fprintf(stderr, "SELECT stmt failed with
return code %d\n", sqlca.sqlcode);
        goto theExit;
       }

   }  /* while <= n (number of orderkeys) */
theExit:
  disconnectDB();
  fclose(fnT);
  fclose(fnQ);
}

/*************** rand_integer *****************/

static int rand_integer ( int val_lo, int val_hi )
{
#ifdef SQLWINT
   return( ((rand()*rand()) % (val_hi-val_lo+1)) +
val_lo );
#else
   return( (random() % (val_hi-val_lo+1)) + val_lo );
#endif
}

/*************** connectDB *****************/

int connectDB(void)
{
   char *dbname_ptr;

   if ((dbname_ptr = getenv("TPCD_QUAL_DBNAME")) !=
NULL )
    {
     strcpy (dbname, getenv("TPCD_QUAL_DBNAME"));
    }
   else
    {
     fprintf(stderr, "TPCD_QUAL_DBNAME not set, using
\"tpcdqual\"");
     strcpy (dbname, "tpcdqual");
    }

/*
EXEC SQL CONNECT TO :dbname;
*/

{
  sqlastrt(sqla_program_id, &sqla_rtinfo, &sqlca);
  sqlaaloc(2,1,3,0L);
    {
      struct sqla_setdata_list sql_setdlist[1];
      sql_setdlist[0].sqltype = 460;
sql_setdlist[0].sqllen = 8;
      sql_setdlist[0].sqldata = (void*)dbname;
      sql_setdlist[0].sqlind = 0L;
      sqlasetdata(2,0,1,sql_setdlist,0L,0L);
    }
  sqlacall((unsigned short)29,4,2,0,0L);
  sqlastop(0L);
}

  if (sqlca.sqlcode != 0)
   {
    fprintf(stderr, "CONNECT TO %s failed with sql
code = %d\n", dbname, sqlca.sqlcode);
    return(1);
   }
  return(0);
} /* end  connectDB */

/*************** disconnectDB *****************/
int disconnectDB(void)
{

/*
EXEC SQL CONNECT RESET;
*/

{
  sqlastrt(sqla_program_id, &sqla_rtinfo, &sqlca);
  sqlacall((unsigned short)29,3,0,0,0L);
  sqlastop(0L);
}

 if (sqlca.sqlcode != 0 )
  {
   fprintf(stderr, "DISCONNECT failed with return code
%d\n", sqlca.sqlcode);
   return(1);
  }
 return(0);
} /* end disconnectDB */
```

mainacid.c

```
/*
########################################################
######################### */
/* ## Licensed Materials - Property of IBM
*/
/* ##
*/
/* ## Governed under the terms of the International
*/
/* ## License Agreement for Non-Warranted Sample Code.
*/
/* ##
*/
/* ## (C) COPYRIGHT International Business Machines
Corp. 1990 - 2005            */
/* ## All Rights Reserved.
*/
/* ##
*/
/* ## US Government Users Restricted Rights - Use,
duplication or                */
/* ## disclosure restricted by GSA ADP Schedule
Contract with IBM Corp.          */
/*
########################################################
######################### */
/*****************************************************
******************/
/*      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>
#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

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[25];
   char   ratefile[25];
   char   outfile[50];
   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_T
MP_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_T
MP_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",gete
nv("TPCD_TMP_DIR"),del());
         log=fopen(buff,"ac");
      }
      else
      {
         sprintf(buff,"%s%cmainacid.log.T",getenv("TPC
D_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);

      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);
         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);

         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);
            #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("TPC
D_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
}
```

tsec.c

```
/*
```

```
##########################################################
######################### */
/* ## Licensed Materials - Property of IBM
*/
/* ##
*/
/* ## Governed under the terms of the International
*/
/* ## License Agreement for Non-Warranted Sample Code.
*/
/* ##
*/
/* ## (C) COPYRIGHT International Business Machines
Corp. 1990 - 2005            */
/* ## All Rights Reserved.
*/
/* ##
*/
/* ## US Government Users Restricted Rights - Use,
duplication or                 */
/* ## disclosure restricted by GSA ADP Schedule
Contract with IBM Corp.           */
/*
##########################################################
######################### */
#include <stdio.h>
#include <time.h>
#include <sys/timeb.h>

/*typedef struct timeval {  */
/*        long    tv_sec;   */      /* seconds */
/*        long    tv_usec;  */      /* and
microseconds */
/*};*/


int main(int argc, char** argv)
{

    struct timeb tb;
    ftime(&tb);
    printf("%us %06uu\n", tb.time, tb.millitm);
    /*printf("%u\n", 1000 * tv.tv_sec +
tv.tv_usec/1000);*/
    return 0;
}
```

makefile

```
##########################################################
######################
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use,
duplication or
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
##########################################################
######################
DBNAME  =       $(TPCD_QUAL_DBNAME)

INCLUDE =       $(HOME)/sqllib/include

#CFLAGS =       -I$(INCLUDE) -g -Dpascal= -DLINT_ARGS \

#LFLAGS  =      -lm -lcurses -ls -ll -ly -liconv -lbsd
EXTRA_CFLAGS=-m64
CFLAGS  =       $(EXTRA_CFLAGS) -I$(INCLUDE) -g
-Dpascal= -DLINT_ARGS  \
                 -DSQLA_NOLINES -DSQLSUN

LFLAGS  =       -lm
# sun .... LFLAGS  =   -lm


LIB     =       -L$(HOME)/sqllib/lib -ldb2

CC      =       cc

HDR     =       acid.h
C       = mainacid.c
SQC     =       acid.sqc
SRC     =       $(HDR)          $(C)            $(SQC)
OBJ     =       acid.o
EXEC    =       mainacid

TARGET  =       $(EXEC) tsec

.SUFFIXES: .o .c .sqc .bnd

.c.o:
        $(CC) -c $< $(CFLAGS)

all:            $(TARGET)

mainacid: $(SRC) $(OBJ) mainacid.o
        $(CC) -o $@ $(CFLAGS) $(OBJ) mainacid.o $(LIB)
$(LFLAGS)

acid.c: acid.sqc  $(HDR)
        - db2 connect to $(DBNAME); \
        db2 prep acid.sqc BINDFILE ISOLATION RR
NOLINEMACRO PACKAGE; \
        db2 bind acid.bnd GRANT PUBLIC; \
        db2 connect reset; \
        db2 terminate

acid.o: acid.c
        $(CC) $(CFLAGS) -c acid.c -o acid.o

tsec: tsec.c
        $(CC) $(CFLAGS) $(LFLAGS) -o tsec tsec.c

clean:
        rm -f *.o *.bnd $(EXEC) tsec
        rm -f acid.c
```

atomicity.pl

```
#!/bin/perl
#
##########################################################
######################
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use,
duplication or
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
##########################################################
######################
#
#$AIX=1;
require "getvars";
require "macro.pl";
require "tpcdmacro.pl";

#if ( !defined $ENV{TPCD_QUAL_DBNAME} )
#{
#  $ENV{TPCD_QUAL_DBNAME} = "tpcd";
#}

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

$dbname=$ENV{"TPCD_QUAL_DBNAME"};
$mode=$ENV{"TPCD_MODE"};

if (( $mode eq "uni" ) || ( $mode eq "smp" ))
{
  $all_ln="once";
```

```
  $all_pn="once";
  $once="once";
}
else
{
  $all_ln="all_ln";
  $all_pn="all_pn";
  $once="once";
}

if ( system("db2 connect to
".$ENV{TPCD_QUAL_DBNAME}) )
{
  print "Database ".$ENV{TPCD_QUAL_DBNAME}." does not
exist. Aborting...\n";
  exit 1;
}

######## ATOMICITY with COMMIT ####################
open(OUTFILE, ">atomc.ver");
chop($timestamp = `perl gettimestamp long`);
print "\n\ncommit atomicity starting at $timestamp\n";
print OUTFILE "commit atomicity starting at
$timestamp\n";
#&rm("/tmp/tpcd/acidT.out.0");
print OUTFILE "        ATOMICITY w/ COMMIT\n";
$tsec = `tsec`;
print OUTFILE "Time = $tsec";
close(OUTFILE);

print "collecting data prior to transaction\n";
#system("db2 -v -t -c -f atom.sql >> atomc.ver 2>&1");
&dodb_conn($dbname,"db2 -v -t -c -f atom.sql >>
atomc.ver 2>&1", $once);
print "Starting ACID TRANSACTION now\n";

#before calling mainacid.exe, in NT, we have to
manually
#export some environment variables.
if ($platform eq "nt")
{
  system("set
TPCD_QUAL_DBNAME=$ENV{$TPCD_QUAL_DBNAME}");
  system("set TPCD_TMP_DIR=$ENV{$TPCD_TMP_DIR}");
}
system("mainacid 2 0 0 113282 1 > xc");

&cat("$ENV{\"TPCD_TMP_DIR\"}${delim}acidT.out.0 >>
atomc.ver");

print "collecting data after transaction has
executed\n";
#system("db2 -v -t -c -f atom.sql >> atomc.ver 2>&1");
&dodb_conn($dbname,"db2 -v -t -c -f atom.sql >>
atomc.ver 2>&1", $once);
&rm("xc $ENV{\"TPCD_TMP_DIR\"}${delim}acidT.out.0");

open(OUTFILE, ">>atomc.ver");
chop($timestamp = `perl gettimestamp long`);
print OUTFILE "commit atomicity finished at
$timestamp\n";
print "commit atomicity finished at $timestamp\n";
close(OUTFILE);

######## ATOMICITY with ROLLBACK ###################
open(OUTFILE, ">atomr.ver");
chop($timestamp = `perl gettimestamp long`);
print "\n\nrollback atomicity starting at
$timestamp\n";
print OUTFILE "rollback atomicity starting at
$timestamp\n";
#&rm("x /tmp/tpcd/acidT.out.0");
print OUTFILE "ATOMICITY w/ ROLLBACK\n";

chop($timestamp = `perl gettimestamp long`);
print OUTFILE "$timestamp\n";
$tsec = `tsec`;
print OUTFILE "Time = $tsec";
close(OUTFILE);

print "collecting data prior to transaction\n";
#system("db2 -v -t -c -f atom.sql >> atomr.ver 2>&1");
&dodb_conn($dbname,"db2 -v -t -c -f atom.sql >>
atomr.ver 2>&1", $once);

print "Starting ACID TRANSACTION now\n";

system("mainacid 2 2 0 113282 1 > xr");
&cat("$ENV{\"TPCD_TMP_DIR\"}${delim}acidT.out.0 >>
atomr.ver");

print "collecting data after transaction has
executed\n";
#system("db2 -v -t -c -f atom.sql >> atomr.ver 2>&1");
&dodb_conn($dbname,"db2 -v -t -c -f atom.sql >>
atomr.ver 2>&1", $once);
&rm("xr $ENV{\"TPCD_TMP_DIR\"}${delim}acidT.out.0");

open(OUTFILE, ">>atomr.ver");
chop($timestamp = `perl gettimestamp long`);
print OUTFILE "rollback atomicity finished at
$timestamp\n";
print "rollback atomicity finished at $timestamp\n";
close(OUTFILE);
#########################################################
#######
if ( -d "$ENV{\"TPCD_AUDIT_DIR\"}" )
{
  &cp("atomc.ver
$ENV{\"TPCD_AUDIT_DIR\"}/auditruns/acid");
  &cp("atomr.ver
$ENV{\"TPCD_AUDIT_DIR\"}/auditruns/acid");
}

system("db2 terminate");

1;
#
```

bind.pl

```
#!/bin/perl
#########################################################
######################
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use,
duplication or
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
#########################################################
######################
#
# bind: binds acid.sqc ... checks for platform nt and
calls nt_complile.bat
#                         or if aix, calls make

# get the environment vars
require "getvars";

# Use the macros in here so that they can handle the
platform differences.
if (length($ENV{"TPCD_QUAL_DBNAME"}) <= 0)
{
  die "TPCD_QUAL_DBNAME environment variable not
set\n";
}
$dbname=$ENV{"TPCD_QUAL_DBNAME"};


if ( $platform eq "nt")
{
 system("nt_compile.bat $dbname");
}
else
{
 system("touch acid.sqc");
 system("make");
}
```

calcrate.pl

```
#!/bin/perl
#########################################################
######################
## Licensed Materials - Property of IBM
```

```
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use,
duplication or
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
#########################################################
#######################
#
#  This script calculates the number of transactions
per second.
#  The input file contains records with "Acid
transaction called at" with
#  the seconds and micoseconds as the next two words.
#
#  We need the first and last transactions times and
the number of transactions
#  to calculate the trans/sec
#$AIX=1;
require "getvars";
require "macro.pl";
require "tpcdmacro.pl";



if ( $#ARGV > -1 )
{
  $infile = $ARGV[0];
}
else
{
  $infile = "consrte.sorted";
}

if ( $#ARGV > 0 )
{
  $numTs = $ARGV[1];
}
else {
  $numTs = 100;
}

if ( ! -r $infile )
{
  print "\n**ERROR**  file $infile could not be
read/found!!!";
  print "           Make sure file $infile exists and
is readable.\n";
  exit 1
}

# parse out the start seconds and microseconds, save
the lowest and highest
# values we find.

$tsFirst = 9999999999;
$tuFirst = 999999;  # NT uses only 3 of these digits.
but it still works.  gav
$tsLast = 0;
$tuLast = 0;

open(INFILE, "<$infile");
while (<INFILE>)
{
# pull out only the transaction lines and get the 5th
and 6th fields.


#  if ( /Acid transaction called at
(\d{9}).\s\d{6}./ )
  if ( /Acid transaction called at
(\d{10})s\s(\d{6})u/ ) #kmw
  {
#    print "acid tx  $1 $2 \n";   #kmw
    if ( ( $tsFirst > $1 )                 ||  # kmw
         ( $tsFirst == $1 && $tuFirst > $2 ) )
    {
      $tsFirst = $1;
      $tuFirst = $2;
    }

    if ( ( $tsLast < $1 )                  ||    # kmw
         ( $tsLast == $1 && $tuLast < $2 ) )
    {
      $tsLast = $1;
      $tuLast = $2;
```

consistency.pl

```
#!/bin/perl
#########################################################
#######################
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use,
duplication or
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
#########################################################
#######################
# Set the dbname if one doesn't already exist.
#if ( ! defined $ENV{TPCD_QUAL_DBNAME} )
#{
#  $ENV{TPCD_QUAL_DBNAME} = "tpcd";
#}

# Set the number of streams if it's not set already.
#if ( ! defined $ENV{TPCD_NUMSTREAM} )
#{
#  $ENV{TPCD_NUMSTREAM} = 3;
#}
require "getvars";
require "macro.pl";
require "tpcdmacro.pl";


if ( $#ARGV > -1 )
{
  $numTs = $ARGV[0];
}
else {
  $numTs = 100;
}

# remove last run's consistency output files
print "removing the last run's consistency output
files\n";

&rm("consrte.*");
&rm("cons*.ver");

$numstreams = $ENV{TPCD_NUMSTREAM} + 1;

# Collect inital consistency data.
print "Running initial consistency checks...\n";
system("perl consist.pl $numTs 1");

# Run the acid transactions for all the query streams
and for the
# update stream as well.
sleep(1);
$timestamp = `perl gettimestamp long`;
print "\n-- Running $numTs ACID Transactions in
$numstreams streams. $timestamp";
system("echo \"-- Running ${numTs} ACID Transactions
in $numstreams streams. $timestamp\" > consrte.out");

# print "-- Running $numTs ACID Transactions in
$numstreams streams. $timestamp" > consrte.out;

# run at least once?

if ($platform eq "nt" )
{
  system("start /b mainacid 3 $numTs 1"); # start /b
works on NT;
}
else
{
```

```
  system("mainacid 3 $numTs 1 &");          # on AIX
submit to background with &
}

sleep(1);

for ( $i = 2 ; $i < $numstreams+1; $i++ )
{

  if ($platform eq "nt") {
     system("start /b mainacid 3 $numTs $i");  # start
/b works on NT;
  } else {
     system("mainacid 3 $numTs $i &");           # on
AIX submit to background with &
  }


  sleep(1);
}

# wait for all the streams for finish
if ($platform eq "nt") {
   sleep(240);
} else {
#    system("sleep 3000");
   sleep(180);
}



for ( $i = 1; $i < $numstreams+1; $i++ )
{
  &cat("$ENV{\"TPCD_TMP_DIR\"}${delim}mainacid.log.T.$
i", "$ENV{\"TPCD_TMP_DIR\"}${delim}mainacid.log.T");
  &rm("$ENV{\"TPCD_TMP_DIR\"}${delim}mainacid.log.T.$i
");
# &cp("$ENV{\"TPCD_TMP_DIR\"}${delim}/rate.$i",
"consrte.out");  # NT did cp, but that captures last
stream only
  &cat("$ENV{\"TPCD_TMP_DIR\"}${delim}rate.$i",
"consrte.out");
  &rm("$ENV{\"TPCD_TMP_DIR\"}${delim}rate.$i");
}

&cp("$ENV{\"TPCD_TMP_DIR\"}${delim}mainacid.log.T",
"$ENV{\"TPCD_AUDIT_DIR\"}${delim}auditexe${delim}acid$
{delim}cons.log");
# sort the output
system("sort < consrte.out > consrte.sorted");
&rm("consrte.out");

# calcuate transaction rates
$totalTXs = $numTs * $numstreams;
system("perl calcrate.pl consrte.sorted $totalTXs >
consrte.ver");

# added this to keep rate calculation on the top of
the file   ..rmy
&cat("consrte.sorted", "consrte.ver");
&rm("consrte.sorted");

# Collect post consistency data.
print "\nRunning post consistency checks...\n";
system("perl consist.pl $numTs 2");
# &rm("$ENV{\"TPCD_TMP_DIR\"}${delim}acid*");

if ( -d "$ENV{TPCD_AUDIT_DIR}" )
{
  &cp("cons1.ver",
"$ENV{TPCD_AUDIT_DIR}/auditruns/acid");
  &cp("consrte.ver",
"$ENV{TPCD_AUDIT_DIR}/auditruns/acid");
  &cp("cons2.ver",
"$ENV{TPCD_AUDIT_DIR}/auditruns/acid");
}

1;
```

consistsql.pl

```
#!/bin/perl
##########################################################
#######################
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use,
duplication or
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
##########################################################
#######################

# Set the platform so the conversion utilities will
know. I'm not sure what
# the $OS is for, we may not need it. chop() removes
newline chars.
#$AIX=1;

# 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";



print " entering consistency.pl\n";
#
# The following setup is specific to the acid runs.
The setup.pl script
# builds the programs and binds to the db.
#

# Set the dbname if one doesn't already exist.
#if ( !defined $ENV{TPCD_QUAL_DBNAME} )
#{
#  $ENV{TPCD_QUAL_DBNAME} = "tpcd";
#}
if (length($ENV{"TPCD_QUAL_DBNAME"}) <= 0)
{
  die "TPCD_QUAL_DBNAME environment variable not
set\n";
}
$dbname=$ENV{TPCD_QUAL_DBNAME};

# numbers in the loop are from the file 10.in which is
generated via the
# gen.input script.
if ( $#ARGV > -1 )
{
  $infile = $ARGV[0];
}
else
{
  $infile = "10.10.in";
}

if ( ! -r $infile )
{
  print "\n**ERROR**  file $infile could not be
read/found!!!";
  print "           Make sure file $infile has 10
lines of";
  print "           L_ORDERKEY values in it.\n";
  exit 1;
}

#if ( system("db2 connect to ".$ENV{TPCD_QUAL_DBNAME})
)
#{
#  print "Cannot connect to".$ENV{TPCD_QUAL_DBNAME}.".
Aborting...\n";
#  exit 1;
#}

open(INFILE, $infile);
while(<INFILE>)
{
  @orderkey = split(' ', $_);
  foreach $orderkey (@orderkey)
  {
```

```
#   &db2("connect to tpcdqual","select
o_orderkey,decimal(o_totalprice,20,2) from tpcd.orders
where o_orderkey = $orderkey","connect
reset","terminate");
    &db2("connect to $dbname","select
o_orderkey,decimal(o_totalprice,20,3) from tpcd.orders
where o_orderkey = $orderkey","select
decimal(sum(decimal(integer(integer(decimal(integer(10
0*decimal(l_extendedprice,20,3)),20,3) * (1-
l_discount)) * (1+l_tax)),20,3)/100.0),20,3) from
tpcd.lineitem where l_orderkey = $orderkey","connect
reset","terminate");
  }
}


#for i in `cat $infile`
#do
#
#echo
"================================================
=============="
#echo
"================================================
=============="
## do not mutiply by 100 on querying the o_totalprice
from tpcd.orders...this
## truncates in the case where our decimail has been
read into a float as .08999999
## instead of .09  as for o_orderkey 23780 (jennifer)
#db2 -v "select o_orderkey,decimal(o_totalprice,20,2)
from tpcd.orders where o_orderkey = $i"
##  db2 -v "select decimal(sum(l_extendedprice*(1-
l_discount)*(1+l_tax)),20,2) from tpcd.lineitem where
l_orderkey = $i"
#
## db2 -v "select
o_orderkey,decimal(integer(100*o_totalprice)/100.0,20,
2) from tpcd.orders where o_orderkey = $i"
## cast the l_extendedprice to decimal first to get
over the float to decimal
## conversion problem (jennifer)  (same problem as
above)
#db2 -v "select
decimal(sum(decimal(integer(integer(decimal(integer(10
0*decimal(l_extendedprice,20,2)),20,2) * (1-
l_discount)) * (1+l_tax)),20,2)/100.0),20,2) from
tpcd.lineitem where l_orderkey = $i"
##db2 -v "select
decimal((sum(decimal(integer(integer(decimal(integer(1
00*l_extendedprice),20,2) * (1-l_discount)) *
(1+l_tax)),20,2)/100.0)),20,2) from tpcd.lineitem
where l_orderkey = $i"
#
##db2 -v "select
sum(integer(100*l_extendedprice)*(100-
integer(l_discount*100))/100*(100+integer(l_tax*100))/
100) from tpcd.lineitem where l_orderkey = $i"
##db2 -v "select
decimal(sum(decimal((((integer(100*l_extendedprice)*(1
00-
integer(l_discount*100)))/100*(100+integer(l_tax*100))
)/100),20,2) / 100.0),20,2) from tpcd.lineitem where
l_orderkey = $i"
#
#done

#sqlcode=`db2 -ec +o "terminate"`;
#if [ $sqlcode != 0 ]
#then
#  echo "terminate failed sqlcode = $sqlcode"
#  exit 1

#fi
print " exiting consistency.pl\n";
```

durab.pl

```
#!/bin/perl
##########################################################
######################
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use,
duplication or
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
##########################################################
######################

# Set the platform so the conversion utilities will
know. I'm not sure what
# the $OS is for, we may not need it. chop() removes
newline chars.
#$AIX=1;

# 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";

#$ENV{"LIBPATH"}="/usr/lib/db2_05_00/lib:/home/audit/s
qllib/lib:/usr/lib:/lib";

if ( $#ARGV < 2 )
{
  print "usage: durab numts name num\n";
  print "where: numts is the number of
transactions.\n";
  print "       name is sym, log or dat\n";
  print "       num is 1 for before or 2 for after\n";
  exit;
}
else
{
  $name = $ARGV[1];
  $num = $ARGV[2];
}

$dbname=$ENV{"TPCD_QUAL_DBNAME"};
$backupDir=$ENV{"TPCD_BACKUP_DIR"};
$mode=$ENV{"TPCD_MODE"};
$platform=$ENV{"TPCD_PLATFORM"};
$db2log=$ENV{"TPCD_DB2LOG"};
$logdir=$ENV{"TPCD_LOG_DIR"};

system("db2start");
system("db2 activate db $dbname");

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";
}

#print "Updating db config for $ENV{TPCD_QUAL_DBNAME}
using logretain yes\n";


if ( $#ARGV > -1 )
{
  $numTs = $ARGV[0];
}
else
{
  $numTs = 100;
}

$numstreams = $ENV{TPCD_NUMSTREAM} + 1;

if ( $num == 1 )
{

# sym test doesn't need to backup db
```

```
if ( ( $name eq "log") || ($name eq "dat") )
{
  print "Deleting all backups in $backupDir!\n";
  &rmdir("$backupDir");
  system("mkdir $backupDir");
  &dodb_noconn("db2 update database configuration for
$dbname using LOGRETAIN yes",$all_ln);
  &dodb_noconn("db2 update database configuration for
$dbname using newlogpath $logdir",$all_ln);
  system("db2stop");

#  if ($platform eq "nt") {
#     system("del $db2log${delim}db2diag.log");
#  } else {
#     system("rm /u/audit/sqllib/db2diag.log");
#  }
  # use system generic version
  &rm("$db2log${delim}db2diag.log");

  system("db2start");

  print "Backing up database $dbname to $backupDir\n";
  if (( $mode eq "mln" ) || ( $mode eq "mpp"))
  {
      # must back up catalog node first...assume node
00
      if ( $platform eq "nt")
      {
          $rc=system("db2_all \"<<+000< db2 backup
database $dbname to $backupDir without prompting\"
");
      }
      else
      {
          $rc=system("db2_all \'}]<<+000< db2 \"backup
database $dbname to $backupDir without prompting\" \'
");
      }
      if ( $rc != 0 )
      {
          die "backup of catalog node failed rc =
$rc\n";
      }
      # back up remaining nodes
      if ( $platform eq "nt" )
      {
          $rc=system("db2_all \"<<-000< db2 backup
database $dbname to $backupDir without prompting\" ");
      }
      elsif ( $platform eq "linux" ){
          $rc=system("db2_all \'}]<<-000< db2 backup
database $dbname to $backupDir without prompting\' ");
      }
      else
      {
          $rc=system("db2_all \'||}]<<-000< db2 backup
database $dbname to $backupDir without prompting\' ");
      }
#    print "skipping the backup temporarily \n";
  }
  else
  {
      $rc=system("db2 -v backup database $dbname to
$backupDir without prompting");
  }

  if ( $rc != 0 )
  {
      die "backup of remaining nodes failed rc =
$rc\n";
  }
  else
  {
      print "BACKUP succesful.\n";
  }
}

  if ( $platform eq "nt")
  {
    print "clearing db2diag.log \n";
    &rm("$db2log${delim}db2diag.log");

  }
  elsif ( $platform eq "aix" )
  {
    print "capturing initial errpt \n";
    &rm("oslg.b4.failure");
    if ( $mode eq "mpp")
    {
      system("rah errpt > oslg.b4.failure");
    }
    else
    {
      system("errpt > oslg.b4.failure");
    }
    print "clearing db2diag.log \n";

    &rm("$db2log${delim}db2diag.log");
  }
  elsif ( $platform eq "sun" )
  {
    &rm("oslg.b4.failure");
    if ( $mode eq "mpp")
    {
      system("rah cat /var/adm/messages >
oslg.b4.failure");
    }
    else
    {
      &cat("/var/adm/messages","oslg.b4.failure");
    }
    &rm("$db2log/db2diag.log");
  }
  elsif ( $platform eq "linux" ){
    &rm("oslg.b4.failure");
    if ( $mode eq "mpp")
    {
      system("rah cat /var/log/messages >
oslg.b4.failure");
    }
    else
    {
      &cat("/var/log/messages","oslg.b4.failure");
    }
    &rm("$db2log/db2diag.log");
  }
  else
  {
    print "durability tests not supported on
$platform. Exiting...\n";
    exit 0;
  }

  print "verifying some ACID properties...\n";
  &rm("$ENV{\"TPCD_TMP_DIR\"}${delim}*");
  do 'history.pl';
  $timestamp = `perl gettimestamp long`;
  system("echo \"-- $timestamp\" >
d${name}cnt$num.ver");
  system("perl durabsql.pl >> d${name}cnt$num.ver");

  $timestamp = `perl gettimestamp long`;
  system("echo \"-- $timestamp \" >
d${name}cons$num.ver");
  system("perl consistsql.pl 10.$numTs.in >>
d${name}cons$num.ver");
  system("perl rundurabT.pl $numTs $name");
}
else
{
  print "Doing post-failure check\n";
  system("touch drate.out");
  &rm("$ENV{\"TPCD_TMP_DIR\"}${delim}mainacid.log.T");
  system("touch
$ENV{\"TPCD_TMP_DIR\"}${delim}mainacid.log.T");
  for ( $i=1; $i < $numstreams+1; $i++ )
  {
    &cat("$ENV{\"TPCD_TMP_DIR\"}${delim}mainacid.log.T
.$i","$ENV{\"TPCD_TMP_DIR\"}${delim}mainacid.log.T");
#
&rm("$ENV{\"TPCD_TMP_DIR\"}${delim}mainacid.log.T.$i")
;
    system("echo \" \" >>
$ENV{\"TPCD_TMP_DIR\"}${delim}rate.$i " );
    &cat("$ENV{\"TPCD_TMP_DIR\"}${delim}rate.$i","drat
e.out");
#   &rm("$ENV{\"TPCD_TMP_DIR\"}${delim}rate.$i");
  }

  $timestamp = `perl gettimestamp long`;
  system("echo \"-- $timestamp\" >
d${name}cnt$num.ver");
```

```
  system("perl durabsql.pl >> d${name}cnt$num.ver");

  $timestamp = `perl gettimestamp long`;
  system("echo \"-- $timestamp\" >
d${name}cons$num.ver");

  system("perl consistsql.pl 10.$numTs.in >>
d${name}cons$num.ver");

  $timestamp = `perl gettimestamp long`;
  system("echo \"-- $timestamp\" > d${name}dblg.ver");

  &cat("$db2log${delim}db2diag.log",
"d${name}dblg.ver");

  $timestamp = `perl gettimestamp long`;
  system("echo \"-- $timestamp\" > d${name}oslg.ver");

  if ( $platform eq "nt" )
  {
      &rm("DURSYS.TXT");
      print "About to run event viewer. You MUST save
the events as DURSYS.TXT\n";
      print "as a TXT file and then exit from the
program.\n";
      print "Waiting 5 seconds so you can read the
above message\n";
      system("eventvwr");
      sleep(15);
      if ( -e "DURSYS.TXT" )
      {
          print "DURSYS.TXT found.\n";
          &cat("DURSYS.TXT", "d${name}oslg.ver");
      }
      else
      {
          print "you did NOT save a DURSYS.TXT\n";
      }
  }
  elsif ( $platform eq "aix" )
  {
    if ( $mode eq "mpp")
    {
      system("rah errpt > oslg.after.failure");
    }
    else
    {
      system("errpt > oslg.after.failure");
    }
    system("diff oslg.b4.failure oslg.after.failure >>
d${name}oslg.ver ");
  }
  elsif ( $platform eq "sun" )
  {
    if ( $mode eq "mpp")
    {
      system("rah cat /var/adm/messages >
oslg.after.failure");
    }
    else
    {
      &cat("/var/adm/messages","oslg.after.failure");
    }
    system("diff oslg.b4.failure oslg.after.failure >>
d${name}oslg.ver ");
  }
  elsif ( $platform eq "linux" ){
    if ( $mode eq "mpp")
    {
      system("rah cat /var/log/messages >
oslg.after.failure");
    }
    else
    {
      &cat("/var/log/messages","oslg.after.failure");
    }
    system("diff oslg.b4.failure oslg.after.failure >>
d${name}oslg.ver ");
  }
  else
  {
    print "SMP/UNI oslg processing not supported on
$platform. Exiting...\n";
    exit 0;
  }

  system("sort < drate.out > drate.sorted");
# &rm("drate.out");

  $countsub=0;
  open(FILE, "drate.sorted");
  while (<FILE>)
  {
    if ( /Acid transaction ./ )
    {
      $countsub++;
    }
  }
  close FILE;

  print "calling calcrate.pl drate.sorted $countsub >
d${name}rate.ver";
  system("perl calcrate.pl drate.sorted $countsub >
d${name}rate.ver");

  &cat("drate.sorted", "d${name}rate.ver");
# &rm("drate.sorted");

  $count2=0;
  open(FILE, "<
$ENV{\"TPCD_TMP_DIR\"}${delim}mainacid.log.T");
  while (<FILE>)
  {
      $count2++;
  }
  close FILE;

  system("echo \"\ncount(*) from
$ENV{\"TPCD_TMP_DIR\"}${delim}mainacid.log.T\" >>
d${name}rate.ver");
  system("echo \"---------------------------------\"
>> d${name}rate.ver");
  system("echo $count2  >> d${name}rate.ver");

  $timestamp = `perl gettimestamp long`;
  system("echo \"-- $timestamp\" > d${name}smpl.ver");

  system("perl gen6q.pl >> d${name}smpl.ver");

  system("echo \" \"  >> d${name}rate.ver");
  system("echo \"entries randomly sampled from success
file: \" >> d${name}rate.ver");
  &cat("6.in"  ,  "d${name}rate.ver" );
# &rm( "6.in" );

  &cp("$ENV{\"TPCD_TMP_DIR\"}${delim}mainacid.log.T",
"$ENV{\"TPCD_AUDIT_DIR\"}${delim}auditexe${delim}acid$
{delim}durab.cons.log");

  if ( $platform eq "nt" )
  {
    # This is nt specific AIX files are written in the
right directory
    if ( -d "$ENV{TPCD_AUDIT_DIR}" )
    {
      print "HERE dir=$ENV{TPCD_AUDIT_DIR}\n";
      &cp("d*.ver",
"$ENV{TPCD_AUDIT_DIR}${delim}auditruns${delim}acid");
    }
  }
}

1;
```

durabsql.pl

```
#!/bin/perl
##########################################################
######################
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use,
duplication or
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
```

```
#########################################################
######################
# Set the platform so the conversion utilities will
know. I'm not sure what
# the $OS is for, we may not need it. chop() removes
newline chars.
#$AIX=1;

# 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";


#if ( system("db2 connect to ".$ENV{TPCD_QUAL_DBNAME})
)
#{
#  print "Cannot connect to
".$ENV{TPCD_QUAL_DBNAME}.". Exiting...\n";
#  exit 0;
#}

$dbname=$ENV{"TPCD_QUAL_DBNAME"};
print "DBNAME= $dbname\n";

#system('db2 select "count(*)" from tpcd.history >
x');
&db2("connect to $dbname","select count(*) from
tpcd.history","connect reset","terminate");

$infile =
"$ENV{\"TPCD_TMP_DIR\"}${delim}mainacid.log.T";
$count2 = 0;


if ( open(INFILE, "<$infile") )
{
  while (<INFILE>)
  {
    $count2++;
  }

  close(INFILE);
}

print "\n";
print "count(*) from $infile\n";
print "-------------------------------------\n";
print "$count2\n";

1;
```

gen6q.pl

```
#!/bin/perl
#########################################################
######################
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use,
duplication or
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
#########################################################
######################
#
# Set the platform so the conversion utilities will
know. I'm not sure what
# the $OS is for, we may not need it. chop() removes
newline chars.
#$AIX=1;

# the following line is supposed to make sure the
redirection
# is flushed.
$| = 1;

require "getvars";
require "macro.pl";
require "tpcdmacro.pl";


#system("db2start");
#if ( system("db2 connect to ".$ENV{TPCD_QUAL_DBNAME})
)
#{
#  print "Database ".$ENV{TPCD_QUAL_DBNAME}." does not
exist. Aborting...\n";
#  exit 1;
#}
if (length($ENV{"TPCD_QUAL_DBNAME"}) <= 0)
{
  die "TPCD_QUAL_DBNAME environment variable not
set\n";
}
$dbname=$ENV{TPCD_QUAL_DBNAME};


$have_compiler =  $ENV{TPCD_HAVECOMPILER};

if ($platform eq "nt") {
  if ($have_compiler eq "yes")
   { system("cl -Zi -DWINNT gen6rnd.c"); }  #   nt
uses cl
}
elsif ( $platform eq "linux" ){
       system("g++ -o gen6rnd gen6rnd.c");
} else {
   system("cc -o gen6rnd gen6rnd.c");
}

$linesinlog=0;
print
"$ENV{\"TPCD_TMP_DIR\"}${delim}mainacid.log.T\n";
open(FILE,
"<$ENV{\"TPCD_TMP_DIR\"}${delim}mainacid.log.T");
while (<FILE>)
{
    $linesinlog++;
}
close FILE;
print "$linesinlog\n";
system("gen6rnd 1 $linesinlog > 6.in.tmp");
system("sort < 6.in.tmp > 6.in");
&rm("6.in.tmp");
print "The 6 random numbers picked are:\n";
&cat("6.in");
print "Line count of success file: $linesinlog\n";

open(NUMFILE, "<6.in");
open(LOGFILE,
"<$ENV{\"TPCD_TMP_DIR\"}${delim}mainacid.log.T");
$reccount = 1;
print
"=================================================
==========";
NUMFILE:
while (<NUMFILE>)
{
  $logrecnum = int($_);
  while(<LOGFILE>)
  {
    if ($reccount eq $logrecnum)
    {
       print "\nRandomly selected entry from
$ENV{\"TPCD_TMP_DIR\"}${delim}mainacid.log.T (line #
$logrecnum)\n ";
      print "\n$_\n";
      if ( /o_key (\d*) / ) { $okey = $1; }
      if ( /l_key (\d*) / ) { $lkey = $1; }
      if ( /delta (\d*) / ) { $delt = $1; }
      if ( /l_partkey (\d*) / ) { $pkey = $1; }
      if ( /l_suppkey (\d*) / ) { $skey = $1; }

      &db2("connect to $dbname","select * from
tpcd.history where h_p_key=$pkey and h_s_key=$skey and
h_o_key=$okey and h_l_key=$lkey and h_delta=$delt
","connect reset","terminate" );
      print
"=================================================
==========\n";
      $reccount++;
      next NUMFILE;
```

```
    }
    $reccount++;
  }
}
close LOGFILE;
close NUMFILE;
print "\n";

# &rm("6.in"); will be removed in durab.pl

#system("db2 terminate");

1;
#
```

gendata.pl

```
#!/bin/perl
##########################################################
######################
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use,
duplication or
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
##########################################################
######################
# gendata.pl durNumT consNumT numQ:
#       generates durNumT random orderkeys and
linenumbers for
#               for the transactions executed as part
of durability test
#       generates numQ orderkeys selected from
durability orderkeys
#               to be used to test consistency before
and after transactions
#               are executed
#       generates consNumT random orderkeys and
linenumbers for
#               for the transactions executed as part
of consistency test
#       generates numQ orderkeys selected from
consistency orderkeys
#               to be used to test consistency before
and after transactions
#               are executed
#
#       typically durNumT=200, consNumT=100 and numQ=10
#
#       for the durability tests, we run
TPCD_NUMSTREAM+1 streams of durNumT
#               transactions each
#       the auditor wants each stream to run a
different set of transactions
#               so we generate TPCD_NUMSTREAMS+1 sets
of orderkeys/linenumbers
#
#       for the consistency tests, we run
TPCD_NUMSTREAM+1 streams of consNumT
#               transactions each
#       the auditor wants each stream to run the same
set of transactions
#               so we generate 1 set of
orderkeys/linenumbers
#               (Note: using the same set of
transactions isn't essential)
#
#       if durNumT=consNumT, the consistency tests will
end up using a
#               different set of transcations for each
stream.
#

# 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";

if ( $#ARGV < 2 )
{
  print "usage: gendata.pl durNumT consNumT numQ\n";
  print "where: durNumT is the number of transactions
for the durability test,\n";
  print "       consNumT is the number of transactions
for the consistency test,\n";
  print "       numQ is the number of queries run as
part of the tests.\n";
  exit;
}

$durNumT = $ARGV[0];
$consNumT = $ARGV[1];
$numQ = $ARGV[2];


$mode = $ENV{TPCD_MODE};
$numstreams = $ENV{TPCD_NUMSTREAM} + 1;
$dbname= $ENV{TPCD_QUAL_DBNAME};
$platform=$ENV{TPCD_PLATFORM};


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";
}

# compile the gendata program
if ( $platform ne "nt" )
{
  if ( $ENV{TPCD_HAVECOMPILER} eq "yes" )
  {
    print "compiling the programs...\n";
    system("make -f Makefile.gendata clean");
    system("make -f Makefile.gendata");
  }
  else
  {
    print "rebinding programs....\n";
    system("db2 connect to ${dbname}");
    system("db2 bind gendata.bnd ISOLATION RR");
    system("db2 connect reset");
  }
}
else
{
  print "on NT, the gendata program should have
compiled succesfully by now\n";
}


# call gendata to create different file
$durNumT.in.$numstream for each stream
# create $numQ.$durNumT.in file to query orderkeys in
one of the streams
# gendata takes orderkeys from 1 to 6000000
for  ($streamNum=1; $streamNum<=$numstreams;
$streamNum++)
{
 system("gendata $durNumT 1 6000000 $streamNum");
}

if (( $durNumT eq $consNumT ))
{
 # consistency and durability tests will use the same
files
 # do nothing
}
else
{
 # call gendata to create one file $consNumT.in.1 and
link a file
 # $consNumT.in.$numstream for each stream from 2 to
$numstream
```

```
 # create $numQ.$consNumT.in file to query orderkeys
in that streams
 system("gendata $consNumT 1 6000000 1");
 for  ($streamNum=2; $streamNum<=$numstreams;
$streamNum++)
 {
 if($platform ne "nt"){
  system("ln -sf ${consNumT}.in.1
${consNumT}.in.${streamNum}");
 }
 else{
  system("copy /Y ${consNumT}.in.1
${consNumT}.in.${streamNum}");
 }
}
}
1;
```

history.pl

```
#!/bin/perl
#########################################################
######################
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use,
duplication or
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
#########################################################
######################
$AIX=1;
require "getvars";
if (length($ENV{"TPCD_QUAL_DBNAME"}) <= 0)
{
  die "TPCD_QUAL_DBNAME environment variable not
set\n";
}
$dbname=$ENV{TPCD_QUAL_DBNAME};

# drop any existing history
&db2("connect to $dbname","drop table tpcd.history",
"connect reset","terminate");
&db2("connect to $dbname",
     "create table TPCD.HISTORY (H_P_KEY INTEGER,
H_S_KEY INTEGER, H_O_KEY INTEGER, H_L_KEY INTEGER,
H_DELTA INTEGER, H_DATE_T TIMESTAMP)",
     "commit", "connect reset",
     "terminate");
```

isolation.pl

```
#!/bin/perl
#########################################################
######################
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use,
duplication or
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
#########################################################
######################
#$AIX=1;
require "getvars";
require "macro.pl";
require "tpcdmacro.pl";


print "**********************\n";
print "* Running ISOLATION1 *\n";
print "**********************\n";
require 'isolation1.pl';
print "**********************\n";
print "* Running ISOLATION2 *\n";
print "**********************\n";
require 'isolation2.pl';
print "**********************\n";
print "* Running ISOLATION3 *\n";
print "**********************\n";
require 'isolation3.pl';
print "**********************\n";
print "* Running ISOLATION4 *\n";
print "**********************\n";
require 'isolation4.pl';
print "**********************\n";
print "* Running ISOLATION5 *\n";
print "**********************\n";
require 'isolation5.pl';
print "**********************\n";
print "* Running ISOLATION6 *\n";
print "**********************\n";
require 'isolation6.pl';
#print "**********************\n";
#print "* Running ISOLATION7 *\n";
#print "**********************\n";
#require 'isolation7.pl';

1;
```

isolation1.pl

```
#!/bin/perl
#########################################################
######################
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use,
duplication or
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
#########################################################
######################
#$AIX=1;

# Set the dbname if one doesn't already exist.
if ( !defined $ENV{TPCD_QUAL_DBNAME} )
{
  $ENV{TPCD_QUAL_DBNAME} = "tpcd";
}

open(OUTFILE, ">iso1.ver");

######## ISOLATION TEST 1 ####################

$timestamp = `perl gettimestamp long`;
print "isolation test #1 starting at $timestamp";
print OUTFILE "isolation test #1 starting at
$timestamp\n";
#&rm("x y /tmp/tpcd/acidT.out.1 /tmp/tpcd/acidQ.out.1
/tmp/tpcd/acidQ.out.11");
print OUTFILE "\tISOLATION TEST 1";
$tsec = `tsec`;
print OUTFILE "\tTime = $tsec";
close(OUTFILE);

# run Txn2 initially to show how it it normally takes.
print "Running an initial Txn2 to show how long it
normally takes. \n";
system("mainacid 1 0 11 1313 > y");

# run acid tranaction Txn1 - request wait before
commit
print "Starting ACID TRANSACTION Txn1 in background;
\n";
print "   it will wait 60 secs before doing COMMIT.
\n";
if ($platform eq "nt") {
   system("start /b mainacid 2 1 1 1313 1 > x"); # NT
uses start /b
} else {
```

```
   system("mainacid 2 1 1 1313 1 > x &");
}



print "Waiting for 10 secs (after which Txn1 should be
waiting) \n";
sleep(10);

print "Starting ACID QUERY Txn2 in background; \n";
print "   it will take much longer the initial Txn2
\n";
print "   because it must wait for Txn1 to
commit/rollback. \n";
if ($platform eq "nt") {
   system("start /b mainacid 1 0 1 1313 > y");  # NT
uses start /b
} else {
   system("mainacid 1 0 1 1313 > y &");
}



# let both Txn's finish
print "Waiting for 90 secs (to let both Txn's finish)
\n";
sleep(90);
&cat("$ENV{\"TPCD_TMP_DIR\"}${delim}acidQ.out.11",
"iso1.ver");
&cat("$ENV{\"TPCD_TMP_DIR\"}${delim}acidT.out.1",
"iso1.ver");
&cat("$ENV{\"TPCD_TMP_DIR\"}${delim}acidQ.out.1",
"iso1.ver");

if ($platform eq "nt") {
   # my NT version of perl does NOT support the wait
function,
   # so I will sleep for 10 seconds and hope that it
does the trick
   # what does that wait do anyways?
   sleep(10);
} else {
   wait;
}

open(OUTFILE, ">>iso1.ver");
$timestamp = `perl gettimestamp long`;
print OUTFILE "isolation test #1 finished at
$timestamp";
close(OUTFILE);
print "isolation test #1 finished at $timestamp \n";

&rm("x y $ENV{\"TPCD_TMP_DIR\"}${delim}acidT.out.1
$ENV{\"TPCD_TMP_DIR\"}${delim}acidQ.out.1
$ENV{\"TPCD_TMP_DIR\"}${delim}acidQ.out.11");
########################################################
#######

if ( -d "$ENV{TPCD_AUDIT_DIR}" )
{
  &cp("iso1.ver",
"$ENV{TPCD_AUDIT_DIR}/auditruns/acid");
}

1;
```

isolation2.pl

```
#!/bin/perl
########################################################
######################
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use,
duplication or
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
########################################################
######################
#$AIX=1;

# Set the dbname if one doesn't already exist.
if ( !defined $ENV{TPCD_QUAL_DBNAME} )
{
  $ENV{TPCD_QUAL_DBNAME} = "tpcd";
}

open(OUTFILE, ">iso2.ver");

######## ISOLATION TEST 2 ####################

$timestamp = `perl gettimestamp long`;
print "isolation test #2 starting at $timestamp";
print OUTFILE "isolation test #2 starting at
$timestamp\n";
#&rm("x y $ENV{\"TPCD_TMP_DIR\"}${delim}acidT.out.2
$ENV{\"TPCD_TMP_DIR\"}${delim}acidQ.out.2
$ENV{\"TPCD_TMP_DIR\"}${delim}acidQ.out.22");
print OUTFILE "\tISOLATION TEST 2";
$tsec = `tsec`;
print OUTFILE "\tTime = $tsec";
close(OUTFILE);

# run Txn2 initially to show how it it normally takes.
print "Running an initial Txn2 to show how long it
normally takes. \n";
system("mainacid 1 0 22 199686 > y");

# run acid tranaction Txn1 - request wait before
rollback
print "Starting ACID TRANSACTION Txn1 in background;
\n ";
print "   it will wait 60 secs before doing ROLLBACK.
\n";
if ($platform eq "nt") {
   system("start /b mainacid 2 3 2 199686 3 > x");  #
NT uses start /b
} else {
   system("mainacid 2 3 2 199686 3 > x &");
}

print "Waiting for 10 secs (after which Txn1 should be
waiting) \n";
sleep(10);

print "Starting ACID QUERY Txn2 in background; \n";
print "   it will take much longer the initial Txn2
\n";
print "   because it must wait for Txn1 to
commit/rollback. \n";
if ($platform eq "nt") {
   system("start /b mainacid 1 0 2 199686 > y"); # NT
uses start /b
} else {
   system("mainacid 1 0 2 199686 > y &");
}

# let both Txn's finish
print "Waiting for 90 secs (to let both Txn's finish)
\n";
sleep(90);

&cat("$ENV{\"TPCD_TMP_DIR\"}${delim}acidQ.out.22",
"iso2.ver");
&cat("$ENV{\"TPCD_TMP_DIR\"}${delim}acidT.out.2",
"iso2.ver");
&cat("$ENV{\"TPCD_TMP_DIR\"}${delim}acidQ.out.2",
"iso2.ver");
&rm("x y $ENV{\"TPCD_TMP_DIR\"}${delim}acidT.out.2
$ENV{\"TPCD_TMP_DIR\"}${delim}acidQ.out.2
$ENV{\"TPCD_TMP_DIR\"}${delim}acidQ.out.22");

open(OUTFILE, ">>iso2.ver");
$timestamp = `perl gettimestamp long`;
print OUTFILE "isolation test #2 finished at
$timestamp";
close(OUTFILE);
print "isolation test #2 finished at $timestamp\n";

########################################################
#######

if ( -d "$ENV{TPCD_AUDIT_DIR}" )
{
```

```
  &cp("iso2.ver",
"$ENV{TPCD_AUDIT_DIR}/auditruns/acid");
}

1;
```

isolation3.pl

```
#!/bin/perl
#########################################################
#######################
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use,
duplication or
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
#########################################################
#######################
#$AIX=1;

# Set the dbname if one doesn't already exist.
if ( !defined $ENV{TPCD_QUAL_DBNAME} )
{
  $ENV{TPCD_QUAL_DBNAME} = "tpcd";
}

open(OUTFILE, ">iso3.ver");

######## ISOLATION TEST 3 ####################
$timestamp = `perl gettimestamp long`;
print "isolation test #3 starting at $timestamp";
print OUTFILE "isolation test #3 starting at
$timestamp\n";
#&rm("x y /tmp/tpcd/acidT.out.3
/tmp/tpcd/acidT.out.33");
print OUTFILE "\tISOLATION TEST 3";
$tsec = `tsec`;
print OUTFILE "\tTime = $tsec";
close(OUTFILE);

# run acid tranaction Txn1 - request wait before
commit
print "Starting ACID TRANSACTION Txn1 in background;
\n";
print "   it will wait 60 secs before doing COMMIT.
\n";

if ($platform eq "nt") {
   system("start /b mainacid 2 1 3 372515 4 > x");  #
NT uses start /b
} else {
   system("mainacid 2 1 3 372515 4 > x &");
}

print "Waiting for 15 secs (after which Txn1 should be
waiting) \n";
sleep(15);

print "Starting ACID TRANSACTION Txn2 in background.
\n";
print "   Without requesting for wait before its
COMMIT, \n";
print "   it is forced to wait for Txn1 to
commit/rollback. \n";
# system("start /b mainacid 2 0 33 1313 4 > y");  # NT
uses start /b
system("mainacid 2 0 33 372515 4 > y &");
#system("mainacid 2 0 33 1313 1 > y &");

# let both Txn's finish
print "Waiting for 90 secs (to let both Txn's finish)
\n";
sleep(90);

&cat("$ENV{\"TPCD_TMP_DIR\"}${delim}acidT.out.3",
"iso3.ver");
&cat("$ENV{\"TPCD_TMP_DIR\"}${delim}acidT.out.33",
"iso3.ver");
&rm("x y $ENV{\"TPCD_TMP_DIR\"}${delim}acidT.out.3
$ENV{\"TPCD_TMP_DIR\"}${delim}acidT.out.33");
$timestamp = `perl gettimestamp long`;

open(OUTFILE, ">>iso3.ver");
print OUTFILE "isolation test #3 finished at
$timestamp";
close(OUTFILE);
print "isolation test #3 finished at $timestamp\n";

#########################################################
#######

if ( -d "$ENV{TPCD_AUDIT_DIR}" )
{
  &cp("iso3.ver",
"$ENV{TPCD_AUDIT_DIR}/auditruns/acid");
}

1;
```

isolation4.pl

```
#!/bin/perl
#########################################################
#######################
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use,
duplication or
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
#########################################################
#######################
#$AIX=1;

# Set the dbname if one doesn't already exist.
if ( !defined $ENV{TPCD_QUAL_DBNAME} )
{
  $ENV{TPCD_QUAL_DBNAME} = "tpcd";
}

open(OUTFILE, ">iso4.ver");

######## ISOLATION TEST 4 ####################
$timestamp = `perl gettimestamp long`;
print "isolation test #4 starting at $timestamp";
print OUTFILE "isolation test #4 starting at
$timestamp \n";
#&rm("x y /tmp/tpcd/acidT.out.4
/tmp/tpcd/acidT.out.44");
print OUTFILE "\tISOLATION TEST 4";

$tsec = `tsec`;
print OUTFILE "\tTime = $tsec";

close(OUTFILE);

# run acid tranaction Txn1 - request wait before
commit
print "Starting ACID TRANSACTION Txn1 in background;
\n";
print "   it will wait 60 secs before doing ROLLBACK.
\n";
if ($platform eq "nt") {
   system("start /b mainacid 2 3 4 67 6 > x"); # NT
uses start /b
} else {
   system("mainacid 2 3 4 67 6 > x &");
}

print "Waiting for 10 secs (after which Txn1 should be
waiting) \n";
sleep(10);

print "Starting ACID TRANSACTION Txn2 in background.
\n";
print "   Without requesting for wait before its
COMMIT, \n";
```

```
print "   it is forced to wait for Txn1 to
commit/rollback. \n";
if ($platform eq "nt") {
   system("start /b mainacid 2 0 44 67 6 > y"); # NT
uses start /b
} else {
   system("mainacid 2 0 44 67 6 > y &");
}

# let both Txn's finish
print "Waiting for 120 secs (to let both Txn's finish)
\n";
sleep(120);

&cat("$ENV{\"TPCD_TMP_DIR\"}${delim}acidT.out.4",
"iso4.ver");
&cat("$ENV{\"TPCD_TMP_DIR\"}${delim}acidT.out.44",
"iso4.ver");
&rm("x y $ENV{\"TPCD_TMP_DIR\"}${delim}acidT.out.4
$ENV{\"TPCD_TMP_DIR\"}${delim}acidT.out.44");
$timestamp = `perl gettimestamp long`;

open(OUTFILE, ">>iso4.ver");
print OUTFILE "isolation test #4 finished at
$timestamp";
close(OUTFILE);
print "isolation test #4 finished at $timestamp\n";

#########################################################
#######

if ( -d "$ENV{TPCD_AUDIT_DIR}" )
{
  &cp("iso4.ver",
"$ENV{TPCD_AUDIT_DIR}/auditruns/acid");
}

1;
```

isolation5.pl

```
#!/bin/perl
#########################################################
######################
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use,
duplication or
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
#########################################################
######################
#$AIX=1;

# Set the dbname if one doesn't already exist.
if ( !defined $ENV{TPCD_QUAL_DBNAME} )
{
  $ENV{TPCD_QUAL_DBNAME} = "tpcd";
}

if ( system("db2 connect to
".$ENV{TPCD_QUAL_DBNAME}) )
{
  print "Database ".$ENV{TPCD_QUAL_DBNAME}." does not
exist. Aborting...\n";
  exit 1;
}

open(OUTFILE, ">iso5.ver");

######## ISOLATION TEST 5 ####################

$timestamp = `perl gettimestamp long`;
print "isolation test #5 starting at $timestamp";
print OUTFILE "isolation test #5 starting at
$timestamp\n";
print OUTFILE "\tISOLATION TEST 5";
$tsec = `tsec`;
print OUTFILE "\tTime = $tsec";

close(OUTFILE);

# check inital state of ORDER, LINEITEM, and HISTORY
print "Determining initial state of ORDER, LINEITEM,
and HISTORY \n";
system("db2 -v -t -c -f iso5chk.sql > iso5.pre 2>&1");

# run acid tranaction Txn1 - request wait before
commit
print "Starting ACID TRANSACTION Txn1 in background;
\n";
print "   it will wait 60 secs before doing COMMIT.
\n";

if ($platform eq "nt") {
   system("start /b mainacid 2 1 5 18439 5 > x"); # NT
uses start /b
} else {
system("mainacid 2 1 5 18439 5 > x &");
}

print "Waiting for 25 secs (after which Txn1 should be
waiting) \n";
sleep(25);

# run PARTSUPP query Txn2 with PS_PARTKEY=1 and
PS_SUPPKEY=2
print "Starting PARTSUPP query Txn2. \n";
system("db2 -v -t -c -f iso5.sql > iso5.out 2>&1");

# wait to ensure Txn1 finishes
print "Waiting for 60 secs (to ensure Txn1 finishes)
\n";
sleep(60);

# check final state of ORDER, LINEITEM, and HISTORY
print "Determining final state of ORDER, LINEITEM, and
HISTORY \n";
system("db2 -v -t -c -f iso5chk.sql > iso5.pst 2>&1");

&cat("iso5.pre", "iso5.ver");
&cat("$ENV{\"TPCD_TMP_DIR\"}${delim}acidT.out.5",
"iso5.ver");
&cat("iso5.out", "iso5.ver");
&cat("iso5.pst", "iso5.ver");
&rm("x y $ENV{\"TPCD_TMP_DIR\"}${delim}acidT.out.5
iso5.pre iso5.out iso5.pst");

open(OUTFILE, ">>iso5.ver");
$timestamp = `perl gettimestamp long`;
print OUTFILE "isolation test #5 finished at
$timestamp";
close(OUTFILE);
print "isolation test #5 finished at $timestamp \n";

#########################################################
#######

if ( -d "$ENV{TPCD_AUDIT_DIR}" )
{
  &cp("iso5.ver",
"$ENV{TPCD_AUDIT_DIR}/auditruns/acid");
}

1;
```

isolation6.pl

```
#!/bin/perl
#########################################################
######################
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use,
duplication or
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
#########################################################
######################
```

```
#$AIX=1;
require "getvars";
require "macro.pl";
require "tpcdmacro.pl";

# Set the dbname if one doesn't already exist.
#if ( !defined $ENV{TPCD_QUAL_DBNAME} )
#{
#  $ENV{TPCD_QUAL_DBNAME} = "tpcd";
#}
if (length($ENV{"TPCD_QUAL_DBNAME"}) <= 0)
{
  die "TPCD_QUAL_DBNAME environment variable not
set\n";
}
$dbname=$ENV{TPCD_QUAL_DBNAME};


if (0> system("db2 activate db
".$ENV{TPCD_QUAL_DBNAME}) )
{
  print "Database ".$ENV{TPCD_QUAL_DBNAME}." does not
exist. Aborting...\n";
  exit 1;
}

if (0> system("db2 connect to
".$ENV{TPCD_QUAL_DBNAME}) )
{
  print "Database ".$ENV{TPCD_QUAL_DBNAME}." does not
exist. Aborting...\n";
  exit 1;
}

$dbname = $ENV{TPCD_QUAL_DBNAME};

open(OUTFILE, ">iso6.ver");

######## ISOLATION TEST 6 ####################

$timestamp = `perl gettimestamp long`;
print "isolation test #6 starting at $timestamp\n";
print OUTFILE "isolation test #6 starting at
$timestamp\n";
print OUTFILE "\tISOLATION TEST 6";

$tsec = `tsec`;
print OUTFILE "\tTime = $tsec";
close(OUTFILE);

# check inital state of ORDER, LINEITEM, and HISTORY
print "Determining initial state of ORDER, LINEITEM,
and HISTORY \n";
system("db2 -v -t -c -f iso6chk.sql > iso6.pre ");

# run Q1 Txn1 with delta=0
#print "Starting Q1 query Txn1 with db2batch. \n";
print "Starting Q1 query Txn1 with db2. \n";

# make sure that db2batch is in the system memory
# so that it loads quickly when we call it below
# we just call it with a dummy sql statement
#system("db2batch -d $dbname -f iso6chk.sql -r
dummy.out -s off ");

&rm("dummy.out");

if ($platform eq "nt") {
   system("start /b db2batch -d $dbname -f iso6.sql1
-r iso6.out1 -s off ");
} else {
# db2batch giving CLI errors switching to db2 cli
#   system("db2batch -d $dbname -f iso6.sql1 -r
iso6.out1 -s off &");
    system("db2 -tvf iso6.sql1 > iso6.out1 &");
}

print "Waiting for some seconds (after which Txn1
should be executing) \n";

# on NT, db2batch takes some time to load, connect and
get things running,
# and get through some of Q1 processing.
# So we give it a 8 second grace
# period...
if ($platform eq "nt")
{
  sleep(8);
}
else
{
  sleep(2);
}

# run acid tranaction Txn2 - request nowait commit
print "Starting ACID TRANSACTION Txn2 in background.
\n";

if ($platform eq "nt") {
   system("start /b mainacid 2 0 6 19169 2 > x &");
} else {
   system("mainacid 2 0 6 19169 2 > x &");
}

# run Q1 Txn3 with delta=2000
print "Starting second Q1, query Txn3, with db2batch.
\n";

if ($platform eq "nt") {
   system("start /b db2batch -d $dbname -f iso6.sql2
-r iso6.out2 -s off ");
} else {
#   system("db2batch -d $dbname -f iso6.sql2 -r
iso6.out2 -s off &");
    system("db2 -tvf iso6.sql2 > iso6.out2 &");
}

# wait to ensure Txn's finishes
print "Waiting for 100 secs (to ensure Txn's finishes)
\n";
sleep(100);

# check final state of ORDER, LINEITEM, and HISTORY
print "Determining final state of ORDER, LINEITEM, and
HISTORY \n";
system("db2 -v -t -c -f iso6chk.sql > iso6.pst ");

system("db2 deactivate db $dbname");

&cat("iso6.pre", "iso6.ver");
&cat("iso6.out1", "iso6.ver");
&cat("$ENV{\"TPCD_TMP_DIR\"}${delim}acidT.out.6",
"iso6.ver");
&cat("iso6.out2", "iso6.ver");
&cat("iso6.pst", "iso6.ver");
&rm("x $ENV{\"TPCD_TMP_DIR\"}${delim}acidT.out.6
iso6.pre iso6.out1 iso6.out2 iso6.pst");

open(OUTFILE, ">>iso6.ver");
$timestamp = `perl gettimestamp long`;
print OUTFILE "isolation test #6 finished at
$timestamp";
close(OUTFILE);
print "isolation test #6 finished at $timestamp \n";

########################################################
#######

if ( -d "$ENV{TPCD_AUDIT_DIR}" )
{
  &cp("iso6.ver",
"$ENV{TPCD_AUDIT_DIR}/auditruns/acid");
}

1;
```

macro.pl

```
#!/usr/bin/perl
########################################################
######################
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use,
duplication or
```

```
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
##########################################################
########################

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

  if ( $OS2 || $WIN )
  {
    $cpcmd = "copy ";
    $cpdircmd = "xcopy ";
    $cpdircmdpreopt = "";
    $cpdircmdpostopt = "/s /e ";
    $catcmd = "type ";
    $rmcmd = "del ";
    $rmcmdopt= "/n";
    $rmdircmd = "rd! ";
  }
  elsif ( $WINT )
  {
    $cpcmd = "copy ";
    $cpdircmd = "xcopy ";
    $cpdircmdpreopt = "";
    $cpdircmdpostopt = "/s /q ";
    $catcmd = "type ";
    $rmcmd = "del ";
    $rmcmdopt= "/q";
    $rmdircmd = "rd /s";
  }
  elsif ($AIX || $UNIX)
  {
    local($PLAT)=&get_uname();

    $cpcmd = "cp ";
    if ( $PLAT eq "SunOS" || $PLAT eq "HP-UX" || $PLAT
=~ "SINIX" ) {
      $cpdircmd = "cp ";
      $cpdircmdpreopt = "-r ";
      $cpdircmdpostopt = "";
    } else {
      $cpdircmd = "cpdir ";
      $cpdircmdpreopt = "";
      $cpdircmdpostopt = "";
    }
    $catcmd = "cat ";
    $rmcmd = "rm -f ";
    $rmcmdopt = "";
    $rmdircmd = "rm -rf ";
  }
  else
  {
    print "invalid operating system" ;
    exit 1;
  }

#==============================
#===  SL4   ===================
#==============================
#==============================

sub sl4 {
  local($src) = @_ ;

  if ( $OS2 || $WIN || $WINT )
  {
    $src =~ s/\\/\\\\/g ;
  }
  return($src);
}

#==============================
#===  SL   ====================
#==============================
#==============================

sub sl {
  local($src) = @_ ;

  if ( $OS2 || $WIN || $WINT )
  {
    $src =~ tr/\//\\/s ;
  }
  return($src);
}

#==============================
#===  CP   ====================
#==============================
#==============================

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

  if ( 0 | -d $src ) {
    -d $dst || mkdir ($dst, 0777) || die "Can't mkdir
$dst: $!\n";
    if ($OS2 || $WIN || $WINT) {
       $src = &sl4( $src );
       $dst = &sl4( $dst );
    }
    system("$cpdircmd $cpdircmdpreopt $src $dst
$cpdircmdpostopt");
  } else {
    system("$cpcmd $src $dst");
  }
}


#==============================
#===  RM   ====================
#==============================
#==============================

sub rm {
  local($src) = @_ ;

    $src =&sl( $src );
    system("$rmcmd $src $rmcmdopt");
}

#==============================
#===  RMDIR  ==================
#==============================
#==============================

sub rmdir {
  local($src) = @_ ;

    system("$rmdircmd $src");
}

#==============================
#===  MV   ====================
#==============================
#==============================

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

    $src =&sl( $src );
    $dst =&sl( $dst );

  if ( $OS2 || $WIN || $WINT )
  {
    system("copy $src $dst ");
    system("del $src");
  }
  else
  {
    system("mv $src $dst");
  }
}

#==============================
#===  CAT  ====================
#==============================
#==============================

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

  $src =&sl( $src );

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

```
  else
  {
     system("$catcmd $src");
  }
}

sub change_codepage {
  local($src)=@_ ;
  if($OS2){
    system("chcp $src");
  }
}



sub get_install_drive {

  if ($OS2) {
     $p=':\\\\os2\\\\install';
  } elsif ($WINT) {
     $p=":\\$ENV{SYSTEMROOT}";
  } else {
     die $noPlatFormMsg;
  }
  ($path, $junk) = split(/$p/io, $ENV{'PATH'}, 2);
  return (substr($path, -1, 1)) ;

}


sub get_uname {

  if ($OS2) {
      $system_name = "OS2";
  }
  elsif ($WIN) {
      $system_name = "WIN";
  }
  elsif ($WINT) {
      $system_name = "WINT";
  }
  else {
      chop($system_name = `uname`);
  }
  $system_name;
}


sub get_dll_path {

  if (($OS2) || ($WIN) || ($WINT)) {
     return ("$ENV{SLAVEDRIVE}\\util") ;
  } else {
     die $noPlatFormMsg;
  }
}

#==============================
# CLIENT-SERVER specifics
#==============================

if (defined $ENV{CLIENT_SERVER})
{

 if ( $AIX )
 {
   $delim="/";
 }
 else
 {
   $delim="\\";
 }

 if ( $AIX_SRV )
 {
   $ssep=' ; ';
   $sset=' export';
   $pfx='. .kshrc;';
   $sdelim='/';
   $srmcmd='rm';
   $scpcmd='cp ';
   $scatcmd='cat';
 }
 else
 {
   $ssep=' & ';
   $sset=' set';
   $pfx=$ENV{SHDRIVE}.': & cd '.$ENV{S_HOME}.' & ';
   $sdelim='\\';
   $srmcmd='echo y | del';
   $scpcmd='copy ';
   $scatcmd='type';
 }

}

#==============================
#===  SSL  ====================
#==============================
#==============================

sub ssl {
  local($src) = @_ ;

  if ( ! $AIX_SRV )
  {
    $src =~ tr/\//\\/s ;
  }
  return($src);
}

#==============================
#===  RSH =====================
#==============================
#==============================

 sub rsh {
  local($RSH_CMD)=@_;

   if (!defined $ENV{CLIENT_SERVER}) {return}
   system("rsh $ENV{'CS_SERVER'} -l $ENV{'USER'}
\"$RSH_CMD\"");
 }

#==============================
#===  GETF ====================
#==============================
#==============================

 sub getf {
  local($rpath) = @_;

   if (!defined $ENV{CLIENT_SERVER}) {return}
   $crpath=$rpath;
   $rpath =&ssl( $rpath );
   $crpath =&sl( $crpath );
   if (! $AIX )
   {
    $crpath =~ tr/\//\\/s ;
   }
   $crpath=$ENV{'TESTDIR'}.$crpath;
   $RSH_CMD=$pfx.$scatcmd." ".$ENV{S_TESTDIR}.$rpath;
   system("rsh $ENV{'CS_SERVER'} -l $ENV{'USER'}
\"$RSH_CMD\" > $crpath");
 }

#==============================
#===  PUTF ====================
#==============================
#==============================

 sub putf {
  local($rpath) = @_;

   if (!defined $ENV{CLIENT_SERVER}) {return}
   local($crpath,$am1,$pfx1);
   $crpath=$rpath;
   $rpath =&sl( $rpath );
   $crpath =&ssl( $crpath );
   if ( $AIX )
   {
    if ( $AIX_SRV )
    { $pfx1='. .kshrc;'; $am1="\'";}
    else
    { $pfx1='. .kshrc\;'; $am1="\"";}
   }
   else
   { $pfx1=''; $am1="\"";}
   $crpath=$ENV{'S_TESTDIR'}.$crpath;
   $rpath=$ENV{TESTDIR}.$rpath;
   if ( $AIX_SRV ) { $rpath =&sl4( $rpath ); }
   $RSH_CMD1=$pfx1.$catcmd." ".$rpath;
```

```
   $RSH_CMD=$pfx."rsh ".$ENV{HOSTNAME}." -l
".$ENV{USER}." ".$am1.$RSH_CMD1.$am1." >".$crpath;
   system("rsh $ENV{'CS_SERVER'} -l $ENV{'USER'}
\"$RSH_CMD\"");
 }

#==============================
#===  DROPDB ==================
#==============================
#==============================

 sub dropdb {
  local($dbalias,$dbname)=@_;

   if (!defined $ENV{CLIENT_SERVER}) {return;}
   if ( $dbname eq "" ) { $dbname=$dbalias; }
   &db2("uncatalog database $dbalias");
   &rsh($pfx."db2 -v drop database ".$dbname);
 }

#==============================
#===  CATNODE =================
#==============================
#==============================

 sub catnode {
  local($node)=@_;

   if (!defined $ENV{CLIENT_SERVER}) {return;}
   &db2("UNCATALOG NODE $node",
        "CATALOG TCPIP NODE $node REMOTE
$ENV{'CS_SERVER'} SERVER $ENV{'S_SERVICE'}" );
   if (! $WIN) {
     &db2("terminate");
   }
 }

#==============================
#===  ATTACH ==================
#==============================
#==============================

 sub attach {
  local($db2cmnd)=@_;
  local($node)="REMNODE";

   if (!defined $ENV{CLIENT_SERVER})
{ &db2("$db2cmnd"); return;}
   &catnode("$node");
   &db2("attach to $node user ".$ENV{USER}." using
".$ENV{USER_PWD},
        "$db2cmnd" );
 }

#==============================
#===  DETACH ==================
#==============================
#==============================

 sub detach {
   if (!defined $ENV{CLIENT_SERVER}) {return;}
   if (! $WIN) {
    &db2("detach");
   }
 }

#==============================
#===  DB2 =====================
#==============================
#==============================
sub db2 {
  local($ldb2opt,$lcmd,@cmd);
  for (@_)
  {
    if    (/^\s*-/) {$ldb2opt = $_;}          # -c -v
-t   ... db2 options
    else            {push(@cmd,$_);}
  }

  if ($ldb2opt!~/p/) {$ldb2opt .= " +p";}     # no
interactive prompt

  $lcmd = "db2 +t -v $ldb2opt";

  $f = "cmd.tmp";
  unlink($f);
  open(F,">$f");
  for (@cmd) {$_.="\n"; print F $_;}
  close(F);
  if ( $WIN )
  {
   $lcmd .= " -f $f";
  }
  else
  {
   $lcmd .= " < $f";
  }
  system($lcmd);
  unlink($f);
}

#==============================
#===  SWITCH_CONFIG ===========
#==============================
#==============================
sub switch_config {
  local($new_config) = @_ ;
  if ($AIX || $UNIX)
  {
    &cp("$ENV{'HOME'}/sqllib/$new_config","$ENV{'HOME'
}/sqllib/db2systm");
  }
  elsif ($OS2)
  {
    &cp("$ENV{DB2PATH}\\$new_config","$ENV{DB2PATH}\\$
ENV{DB2INSTANCE}\\db2systm");
    &cp("$ENV{DB2PATH}\\$new_config","$ENV{DB2PATH}\\$
ENV{DB2INSTANCE}\\db2syst");
  }
  elsif ($WINT)
  {
    &cp("$ENV{DB2PATH}\\$new_config","$ENV{DB2PATH}\\$
ENV{DB2INSTANCE}\\db2systm");
  }
  return;
}

#==============================
#===  QUERY_LANG ==============
#==============================
#==============================
sub query_LANG {
  local($PLAT)=&get_uname();

  if ($PLAT eq "AIX" || $OS2) {
     return $ENV{LANG};
  } elsif ($PLAT eq "SunOS") {
     if ($ENV{LANG} eq "en_US") {
        return "En_US";
     }
     if ($ENV{LANG} eq "de") {
        return "da_DK";
     }
  } elsif ($PLAT eq "HP-UX") {
     if ($ENV{LANG} eq "en_US.iso88591") {
        return "En_US";
     }
     if ($ENV{LANG} eq "de_DE.iso88591") {
        return "da_DK";
     }
  }

}
#==============================
#===  LANG_EXISTS =============
#==============================
#==============================
sub LANG_exists {
  local($src) = @_ ;
  local($PLAT)=&get_uname();

  if ($PLAT eq "AIX" || $OS2) {
     return(1);
  } elsif ($PLAT eq "SunOS" || $PLAT eq "HP-UX" ||
$PLAT =~ "SINIX") {
     if ($src ne "En_US" && $src ne "da_DK" && $src ne
"fr_FR"
       && $src ne "es_ES" && $src ne "it_IT" && $src
ne "ko_KR") {
        return(0);
     }
     return(1);
```

```
  }

}

#==============================
#===  SET_LANG ================
#==============================
#==============================
sub set_LANG {
  local($src) = @_ ;
  local($PLAT)=&get_uname();

  if ($PLAT eq "AIX" || $OS2) {
     $ENV{'LANG'}=$src;
  } elsif ($PLAT eq "SunOS") {
     if ( &LANG_exists($src) ) {
        if ($src eq "En_US") {
           $ENV{'LANG'}="en_US";
        } elsif ($src eq "da_DK") {
           $ENV{'LANG'}="de";
        } elsif ($src eq "fr_FR") {
           $ENV{'LANG'}="fr";
        } elsif ($src eq "es_ES") {
           $ENV{'LANG'}="es";
        } elsif ($src eq "it_IT") {
           $ENV{'LANG'}="it";
        } elsif ($src eq "ko_KR") {
           $ENV{'LANG'}="ko";
        } else {
           $ENV{'LANG'}=$src;
        }
     } else {
        return(0);
     }
  } elsif ($PLAT eq "HP-UX") {
     if ( &LANG_exists($src) ) {
        if ($src eq "En_US") {
           $ENV{'LANG'}="en_US.iso88591";
        } elsif ($src eq "da_DK") {
           $ENV{'LANG'}="de_DE.iso88591";
        } elsif ($src eq "fr_FR") {
           $ENV{'LANG'}="fr_FR.iso88591";
        } elsif ($src eq "es_ES") {
           $ENV{'LANG'}="es_ES.iso88591";
        } elsif ($src eq "it_IT") {
           $ENV{'LANG'}="it_IT.iso88591";
        } elsif ($src eq "ko_KR") {
           $ENV{'LANG'}="ko_KR.eucKR";
        } else {
           $ENV{'LANG'}=$src;
        }
     } else {
        return(0);
     }
  } elsif ($PLAT =~ "SINIX") {
     if ( &LANG_exists($src) ) {
        if ($src eq "En_US") {
           $ENV{'LANG'}="En";
        } elsif ($src eq "da_DK") {
           $ENV{'LANG'}="De";
        } elsif ($src eq "fr_FR") {
           $ENV{'LANG'}="Fr";
        } else {
           $ENV{'LANG'}=$src;
        }
     } else {
        return(0);
     }
  }

  return(1);
}

#==============================
#===  SET_LC_ALL ==============
#==============================
#==============================
sub set_LC_ALL {
  local($src) = @_ ;
  local($PLAT)=&get_uname();

  if ($PLAT eq "AIX" || $OS2) {
     $ENV{'LC_ALL'}=$src;
  } elsif ($PLAT eq "SunOS") {
     if ( &LANG_exists($src) ) {
        if ($src eq "En_US") {
           $ENV{'LC_ALL'}="en_US";
        } elsif ($src eq "da_DK") {
           $ENV{'LC_ALL'}="de";
        } elsif ($src eq "fr_FR") {
           $ENV{'LC_ALL'}="fr";
        } elsif ($src eq "es_ES") {
           $ENV{'LC_ALL'}="es";
        } elsif ($src eq "it_IT") {
           $ENV{'LC_ALL'}="it";
        } else {
           $ENV{'LC_ALL'}=$src;
        }
     } else {
        return(0);
     }
  } elsif ($PLAT eq "HP-UX") {
     if ( &LANG_exists($src) ) {
        if ($src eq "En_US") {
           $ENV{'LC_ALL'}="en_US.iso88591";
        } elsif ($src eq "da_DK") {
           $ENV{'LC_ALL'}="de_DE.iso88591";
        } elsif ($src eq "fr_FR") {
           $ENV{'LC_ALL'}="fr_FR.iso88591";
        } elsif ($src eq "es_ES") {
           $ENV{'LC_ALL'}="es_ES.iso88591";
        } elsif ($src eq "it_IT") {
           $ENV{'LC_ALL'}="it_IT.iso88591";
        } else {
           $ENV{'LC_ALL'}=$src;
        }
     } else {
        return(0);
     }
  } elsif ($PLAT =~ "SINIX") {
     if ( &LANG_exists($src) ) {
        if ($src eq "En_US") {
           $ENV{'LC_ALL'}="En";
        } elsif ($src eq "da_DK") {
           $ENV{'LC_ALL'}="De";
        } elsif ($src eq "fr_FR") {
           $ENV{'LC_ALL'}="Fr";
        } else {
           $ENV{'LC_ALL'}=$src;
        }
     } else {
        return(0);
     }
  }recover.

  return(1);
}

#==============================
#===  SET_LC_CTYPE ============
#==============================
#==============================
sub set_LC_CTYPE {
  local($src) = @_ ;
  local($PLAT)=&get_uname();

  if ($PLAT eq "AIX" || $OS2) {
     $ENV{'LC_CTYPE'}=$src;
  } elsif ($PLAT eq "SunOS") {
     if ( &LANG_exists($src) ) {
        if ($src eq "En_US") {
           $ENV{'LC_CTYPE'}="en_US";
        } elsif ($src eq "da_DK") {
           $ENV{'LC_CTYPE'}="de";
        } elsif ($src eq "fr_FR") {
           $ENV{'LC_CTYPE'}="fr";
        } elsif ($src eq "es_ES") {
           $ENV{'LC_CTYPE'}="es";
        } elsif ($src eq "it_IT") {
           $ENV{'LC_CTYPE'}="it";
        } elsif ($src eq "ko_KR") {
           $ENV{'LC_CTYPE'}="ko";
        } else {
           $ENV{'LC_CTYPE'}=$src;
        }
     } else {
        return(0);
     }
  } elsif ($PLAT eq "HP-UX") {
     if ( &LANG_exists($src) ) {
        if ($src eq "En_US") {
           $ENV{'LC_CTYPE'}="en_US.iso88591";
```

```
        } elsif ($src eq "da_DK") {
           $ENV{'LC_CTYPE'}="de_DE.iso88591";
        } elsif ($src eq "fr_FR") {
           $ENV{'LC_CTYPE'}="fr_FR.iso88591";
        } elsif ($src eq "es_ES") {
           $ENV{'LC_CTYPE'}="es_ES.iso88591";
        } elsif ($src eq "it_IT") {
           $ENV{'LC_CTYPE'}="it_IT.iso88591";
        } elsif ($src eq "ko_KR") {
           $ENV{'LC_CTYPE'}="ko_KR.eucKR";
        } else {
           $ENV{'LC_CTYPE'}=$src;
        }
     } else {
        return(0);
     }
  } elsif ($PLAT =~ "SINIX") {
     if ( &LANG_exists($src) ) {
        if ($src eq "En_US") {
           $ENV{'LC_CTYPE'}="En";
        } elsif ($src eq "da_DK") {
           $ENV{'LC_CTYPE'}="De";
        } elsif ($src eq "fr_FR") {
           $ENV{'LC_CTYPE'}="Fr";
        } else {
           $ENV{'LC_CTYPE'}=$src;
        }
     } else {
        return(0);
     }
  }

  return(1);
}

1;
```

recover.pl

```
#!/bin/perl
##########################################################
######################
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use,
duplication or
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
##########################################################
######################

# Set the platform so the conversion utilities will
know. I'm not sure what
# the $OS is for, we may not need it. chop() removes
newline chars.
#$AIX=1;

# 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";


if ( $#ARGV != 0 )
{
  print "usage: recover name\n";
  print "where: name is sym, log or dat\n";
  exit;
}
else
{
  $name = $ARGV[0];
}

$dbname=$ENV{TPCD_QUAL_DBNAME};
$backupDir=$ENV{TPCD_BACKUP_DIR};
$mode=$ENV{TPCD_MODE};
$platform=$ENV{TPCD_PLATFORM};

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 ( $name eq "dat" )
{
  print "performing restore database on each node, and
then roll forward ... \n";  }
system("sleep 10");
system("db2 rollforward database $dbname to end of
logs and stop" );
}
else
{
  print "usage: recover name\n";
  print "where: name is sym, log or dat\n";
}

print
"================================================
\n";
print "recovery procedures for failure type $name
completed. \n";
print "you can proceed with part2 of the durability
tests.   \n";
print
"================================================
\n";

1;
```

runaci.pl

```
#!/bin/perl
##########################################################
######################
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use,
duplication or
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
##########################################################
######################
#
# runaci.pl: Run the tpcd acid tests. This is the main
perl script.
#            It is assumed you have a database named
either tpcf or
#            defined in TPCD_QUAL_DBNAME.
#
# change history:
# Feb 13, 1996 - Converted from ksh to perl and
standardized for cross platform
#

# Set the platform so the conversion utilities will
know. I'm not sure what
# the $OS is for, we may not need it. chop() removes
newline chars.
#$AIX=1;


#  jel add libbsd.a to linker libs to retrieve ftime
#      I *think* it would be safer to specify libc.a
as well but
#      that gets brought in from some other export so
leave it for now
```

```
#$C_FLAGS="-ls -liconv -lm -lXm -lbsd -Dpascal=
-D_loadds= -Dfar= -DSQLUNIX -DSQLAIX -DLINT_ARGS
-DSQLZDEBUG -DSQLTOAIX";
#$ENV{'C_FLAGS'} = $C_FLAGS;

print "STARTING runACI.pl\n";
# get the environment vars
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";

# tpcdmacro has commands for running db2 commands (eg,
dodb_conn)
require "tpcdmacro.pl";

# if uni then only do the db commands once ...
otherwise per node
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";
}


print "VARIABLES LOADED\n";


if ( $platform eq "nt")
{
   print "library path set in nt_compile.bat\n";
}
else
{
   $ENV{"LIBPATH"}="/usr/lib/db2_05_00/lib:/home/audit
/sqllib/lib:/usr/lib:/lib";
}

#
# The following setup is specific to the acid runs.
The setup.pl script
# builds the programs and binds to the db.
#

# Setup the tmp dir used by the runs
if ( ! -d $ENV{"TPCD_TMP_DIR"} )
{
  $rc = mkdir($ENV{"TPCD_TMP_DIR"},0777);
  print("creating the temp dir, rc = $rc\n");
}

$dbname=$ENV{"TPCD_QUAL_DBNAME"};
print "DBNAME= $dbname\n";

# activate the database
&dodb_noconn( "db2 activate db $dbname" , $once );

# Check if there are summary Tables created
#print "The following Summary Tables exist:\n";
#&dodb_conn("$dbname",
           #"db2 \"select name from sysibm.systables
where type='S'\"",
           #$once);



# Delete any existing stuff in tmp. The rm macro does
a -f by default.
&rm($ENV{"TPCD_TMP_DIR"}."${delim}*");

# if a compiler is available, and we are on NT, then
recompile
# the C files
if ($platform eq "nt")
{
  if ($ENV{"TPCD_HAVECOMPILER"} eq "yes")
  {
     printf("compiling mainacid.c and acid.c
.........\n");
     system("nt_compile $dbname > nt_compile.out");
     printf("compiling tsec.c  .........\n");
system("nt_tsec_compile > nt_tsec_compile.out");
  }
  else
  {
     print(" ERROR>>>>>>> you are on NT, and you will
need \n");
     print("              a compiler, or find
binaries.\n");
     print("              check nt_compiler.bat to
see\n");
  }
}


print "\nPerforming inital setup...\n";
require 'setup.pl';

print "\nRunning atomicity tests...\n";
require 'atomicity.pl';

print "\nRunning consistency tests...\n";
require 'consistency.pl';

print "\nRunning isolation tests...\n";
require 'isolation.pl';

print "Cleaning up the temp directory... \n";

&rm("$ENV{\"TPCD_TMP_DIR\"}${delim}*");
print "runACI.pl is finished.\n";
```

setup.pl

```
#!/bin/perl
##########################################################
#######################
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use,
duplication or
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
##########################################################
#######################
#
# setup.pl: create the history table and compile all
programs.
#$AIX=1;

require "getvars";
require "macro.pl";

# set some environment vars for running rtest.
#chop( $TESTDIR=`pwd` );
#if ( !defined $ENV{TESTDIR} )
#{
#  $ENV{TESTDIR} = "F:/tpcdaudit/acid";
#}

print "creating history table...\n";
if ( system("db2 connect to
".$ENV{TPCD_QUAL_DBNAME}) )
{
  print "Database ".$ENV{TPCD_QUAL_DBNAME}." does not
exist. Aborting...\n";
  exit 1;
}
do 'history.pl';

#
# This should really be in a macro!
#
if ( $platform eq "nt" )
{
```

```
    $sep="&";
}
else
{
    $sep=";";
}

if ( $platform ne "nt" )
{
  if ( $ENV{TPCD_HAVECOMPILER} eq "yes" )
  {
    print "compiling the programs...\n";
    system("make clean");
    system("make ");
  }
  else
  {
    print "rebinding programs....\n";
    system("db2 connect to ".$ENV{TPCD_QUAL_DBNAME});
    system("db2 bind acid.bnd ISOLATION RR");
    system("db2 connect reset");
  }
}
else
{
  print "on NT, the programs should have compiled
succesfully by now\n";
}

# setup for the main() compile.
#$ENV{'C_OBJS'}="acid.o";
#$ENV{'C_FLAGS'}="-Dpascal= -DSQLUNIX -DSQLAIX
-DLINT_ARGS -Dfar= -D_loadds= -DSQLA_NOLINES -qflag=i:
i -qlanglvl=ansi";

1;
```

tpcdmacro.pl

```
#!/usr/bin/perl
#########################################################
######################
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use,
duplication or
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
#########################################################
######################
# 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 "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 "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 "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);
}

sub outtime
{
    local($stmt) = @_;
    local($current_time) = 0;

    $current_time = `perl gettimestamp long`;
    print "$stmt at : $current_time\n ";
}
1;
```

watchdurab.pl

```
#!/bin/perl
#########################################################
######################
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use,
duplication or
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
#########################################################
######################
# usage  watchdurab xxx
#        where xxx = the type of failure (used for
naming the output file)
#  NOTE:  once the failure has occurred, the
durab.control file must be
#         touched or created or this program will
continue to run!
require 'getvars';
require 'macro.pl';
require 'tpcdmacro.pl';

#---- Initialize variables #
#timestampmon=`date +%y%m%d-3/28/07M%S`

# Set the dbname if one doesn't already exist.
#if ( !defined $ENV{TPCD_QUAL_DBNAME} )
#{
#  $ENV{TPCD_QUAL_DBNAME} = "tpcdqual";
#}

if ( $#ARGV > -1 )
{
  $name = $ARGV[0];
}

#: ${TPCD_RUNNUMBER:?"TPCD_RUNNUMBER environment
variable not set"}
#: ${TPCD_RUN_DIR:?"TPCD_RUN_DIR environment variable
not set"}

#first cleanup the old "synchronization" fileif ( -f
"durab.control" )
{
  &rm("durab.control");
}

$curdate = `perl gettimestamp long`;
system("echo \"Started monitoring the streams (60
second interval) starting at : $curdate\" >
d${name}appl.ver");
# run this until the last file produced by the
throughput streams has
# been created.  At this point all the update and
query streams are finished
$count = 0;
print "$count";
while ( ! -f "durab.control" )
{
  $timestamp = `perl gettimestamp long`;
  system("echo \"$timestamp\" >> d${name}appl.ver");
  system("db2 list applications for database
$ENV{TPCD_QUAL_DBNAME} >> d${name}appl.ver");
  $timestamp = `perl gettimestamp long`;
  system("echo \"$timestamp\" >> d${name}appl.ver");
  # sleep for 60 seconds before we do this again
  sleep(60);
  $count++;
  print "$count";
  if ( $count == 10 )
  {
    $timestamp = `perl gettimestamp long`;
    system("echo \"Completed monitoring the streams at
: $timestamp\" >> d${name}appl.ver");
    exit;
  }
}

$timestamp = `perl gettimestamp long`;
system("echo \"Completed monitoring the streams at :
$timestamp\" >> d${name}appl.ver");

1;
```

==========

Disk Configuration Details

Basic Assumptions:

- Segregate logs from other database objects
- Configure for 4 database partitions per physical node
- Use Solaris Volume Manager (SVM) for volume management

Disk Config per node:

- 2-way mirror SVM metadevice for logs
- 2-way mirror SVM metadevice for system tablespace
- 11-way RAID10 metadevices for each database partition
- SVM soft partitions to create containers for tablespaces

Sample Disk Configuration:

```
metainit d11 1 11 c0t2d0s0 c1t2d0s0 c4t2d0s0 c5t2d0s0
c6t2d0s0 c7t2d0s0 c0t1d0s0 c4t7d0s0 c1t0d0s0 c7t4d0s0
c6t1d0s0 -i 128k
metainit d12 1 11 c7t3d0s4 c6t3d0s4 c5t3d0s4 c4t3d0s4
c1t3d0s4 c0t3d0s4 c1t1d0s4 c5t7d0s4 c4t0d0s4 c6t4d0s4
c7t1d0s4 -i 128k

metainit d1 -m d11 d12 -r

# DATA_INDEX tablespace
metainit d100 -p d1 150g
# TEMP32K tablespace
metainit d101 -p d$i 510g

metainit d81 1 1 c0t0d0s0
metainit d82 1 1 c0t4d0s0
metainit d8 -m d81 d82

# System Tablespace
metainit d800 -p d8 445g

# LOGS
metainit d801 -p d8 5g
metainit d802 -p d8 5g
metainit d803 -p d8 5g
metainit d804 -p d8 5g
```

Appendix C. Query Text and Query Output

```
================================================
```

qryqual01

```
================================================
Start timestamp 09/28/07 10:36:14.073047

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

-- 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
56586554400.729       53758257134.869
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
111701729697.743      106118230307.606
110367043872.499               25.502
38249.118                  0.050      2920374
R             F                     37719753.000
56568041380.899       53741292684.604
55889619119.832                25.506
38250.855                  0.050      1478870


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


Stop timestamp 09/28/07 10:36:24.768082
Query Time =            10.7 secs

================================================
```

qryqual02

```
================================================
Start timestamp 09/28/07 10:30:15.445861

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

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

Tag: Q2     Stream: -1   Sequence number: 2

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

S_ACCTBAL             S_NAME
N_NAME                     P_PARTKEY    P_MFGR
S_ADDRESS                               S_PHONE
S_COMMENT
------------------------------------------------------
------------------------------------------------------
------------------------------------------------------
------------------------------------------------------
------------------------------------------------------
--------
          9938.530  Supplier#000005359
UNITED KINGDOM                185358  Manufacturer#4
QKuHYh,vZGiwu2FWEJoLDx04                33-429-790-
6131  uriously regular requests hag
          9937.840  Supplier#000005969
ROMANIA                       108438  Manufacturer#1
ANDENSOSmk,miq23Xfb5RWt6dvUcvt6Qa       29-520-692-
3537  efully express instructions. regular requests
against the slyly fin
          9936.220  Supplier#000005250
UNITED KINGDOM                   249  Manufacturer#4
B3rqp0xbSEim4Mpy2RH J                   33-320-228-
2957  etect about the furiously final accounts. slyly
ironic pinto beans sleep inside the furiously
          9923.770  Supplier#000002324
GERMANY                        29821  Manufacturer#4
y3OD9UywSTOk                            17-779-299-
1839  ackages boost blithely. blithely regular
deposits c
          9871.220  Supplier#000006373
GERMANY                        43868  Manufacturer#5
J8fcXWsTqM                              17-813-485-
8637  etect blithely bold asymptotes. fluffily ironic
platelets wake furiously; blit
          9870.780  Supplier#000001286
GERMANY                        81285  Manufacturer#2
YKA,E2fjiVd7eUrzp2Ef8j1QxGo2DFnosaTEH    17-516-924-
4574  regular accounts. furiously unusual courts above
the fi
          9870.780  Supplier#000001286
```

```
GERMANY                         181285  Manufacturer#4
YKA,E2fjiVd7eUrzp2Ef8j1QxGo2DFnosaTEH     17-516-924-
4574  regular accounts. furiously unusual courts above
the fi
          9852.520  Supplier#000008973
RUSSIA                           18972  Manufacturer#2
t5L67YdBYYH6o,Vz24jpDyQ9                  32-188-594-
7038  rns wake final foxes. carefully unusual depende
          9847.830  Supplier#000008097
RUSSIA                          130557  Manufacturer#2
xMe97bpE69NzdwLoX                         32-375-640-
3593  the special excuses. silent sentiments serve
carefully final ac
          9847.570  Supplier#000006345
FRANCE                           86344  Manufacturer#1
VSt3rzk3qG698u6ld8HhOByvrTcSTSvQlDQDag    16-886-766-
7945  ges. slyly regular requests are. ruthless,
express excuses cajole blithely across the unu
          9847.570  Supplier#000006345
FRANCE                          173827  Manufacturer#2
VSt3rzk3qG698u6ld8HhOByvrTcSTSvQlDQDag    16-886-766-
7945  ges. slyly regular requests are. ruthless,
express excuses cajole blithely across the unu
          9836.930  Supplier#000007342
RUSSIA                           4841  Manufacturer#4
JOlK7C1,7xrEZSSOw                         32-399-414-
5385  blithely carefully bold theodolites. fur
          9817.100  Supplier#000002352
RUSSIA                          124815  Manufacturer#2
4LfoHUZjgjEbAKw TgdKcgOc4D4uCYw           32-551-831-
1437  wake carefully alongside of the carefully final
ex
          9817.100  Supplier#000002352
RUSSIA                          152351  Manufacturer#3
4LfoHUZjgjEbAKw TgdKcgOc4D4uCYw           32-551-831-
1437  wake carefully alongside of the carefully final
ex
          9739.860  Supplier#000003384
FRANCE                          138357  Manufacturer#2
o,Z3v4POifevE k9U1b 6J1ucX,I              16-494-913-
5925  s after the furiously bold packages sleep
fluffily idly final requests: quickly final
          9721.950  Supplier#000008757
UNITED KINGDOM                  156241  Manufacturer#3
Atg6GnM4dT2                               33-821-407-
2995  eep furiously sauternes; quickl
          9681.330  Supplier#000008406
RUSSIA                           78405  Manufacturer#1
,qUuXcftUl                                32-139-873-
8571  haggle slyly regular excuses. quic
          9643.550  Supplier#000005148
ROMANIA                         107617  Manufacturer#1
kT4ciVFslx9z4s79p Js825                   29-252-617-
4850  final excuses. final ideas boost quickly
furiously speci
          9624.820  Supplier#000001816
FRANCE                           34306  Manufacturer#3
e7vab91vLJPWxxZnewmnDBpDmxYHrb            16-392-237-
6726  e packages are around the special ideas.
special, pending foxes us
          9624.780  Supplier#000009658
ROMANIA                         189657  Manufacturer#1
oE9uBgEfSS4opIcepXyAYM,x                  29-748-876-
2014  ronic asymptotes wake bravely final
          9612.940  Supplier#000003228
ROMANIA                         120715  Manufacturer#2
KDdpNKN3cWu7ZSrbdqp7AfSLxx,qWB            29-325-784-
8187  warhorses. quickly even deposits sublate
daringly ironic instructions. slyly blithe t
          9612.940  Supplier#000003228
ROMANIA                         198189  Manufacturer#4
KDdpNKN3cWu7ZSrbdqp7AfSLxx,qWB            29-325-784-
8187  warhorses. quickly even deposits sublate
daringly ironic instructions. slyly blithe t
          9571.830  Supplier#000004305
ROMANIA                         179270  Manufacturer#2
qNHZ7WmCzygwMPRDO9Ps                      29-973-481-
1831  kly carefully express asymptotes. furiou
          9558.100  Supplier#000003532
UNITED KINGDOM                   88515  Manufacturer#4
EOeuiiOn21OVpTlGguufFDFsbN1p0lhpxHp       33-152-301-
2164  foxes. quickly even excuses use. slyly special
foxes nag bl
          9492.790  Supplier#000005975
GERMANY                          25974  Manufacturer#5
S6mIiCTx82z7lV                            17-992-579-
4839  arefully pending accounts. blithely regular
excuses boost carefully carefully ironic p
          9461.050  Supplier#000002536
UNITED KINGDOM                   20033  Manufacturer#1
8mmGbyzaU 7ZS2wJumTibypncu9pNkDc4FYA      33-556-973-
5522  . slyly regular deposits wake slyly. furiously
regular warthogs are.
          9453.010  Supplier#000000802
ROMANIA                         175767  Manufacturer#1
,6HYXb4uaHITmtMBj4Ak57Pd                  29-342-882-
6463  gular frets. permanently special multipliers
believe blithely alongs
          9408.650  Supplier#000007772
UNITED KINGDOM                  117771  Manufacturer#4
AiC5YAH,gdu0i7                            33-152-491-
1126  nag against the final requests. furiously
unusual packages cajole blit
          9359.610  Supplier#000004856
ROMANIA                          62349  Manufacturer#5
HYogcF3Jb yh1                             29-334-870-
9731  y ironic theodolites. blithely sile
          9357.450  Supplier#000006188
UNITED KINGDOM                  138648  Manufacturer#1
g801,ssP8wpTk4Hm                          33-583-607-
1633  ously always regular packages. fluffily even
accounts beneath the furiously final pack
          9352.040  Supplier#000003439
GERMANY                         170921  Manufacturer#4
qYPDgoiBGhCYxjgC                          17-128-996-
4650  according to the carefully bold ideas
          9312.970  Supplier#000007807
RUSSIA                           90279  Manufacturer#5
oGYMPCk9XHGB2PBfKRnHA                     32-673-872-
5854  ecial packages among the pending, even requests
use regula
          9312.970  Supplier#000007807
RUSSIA                          100276  Manufacturer#5
oGYMPCk9XHGB2PBfKRnHA                     32-673-872-
5854  ecial packages among the pending, even requests
use regula
          9280.270  Supplier#000007194
ROMANIA                          47193  Manufacturer#3
zhRUQkBSrFYxIAXTfInj vyGRQjeK             29-318-454-
2133  o beans haggle after the furiously unusual
deposits. carefully silent dolphins cajole carefully
          9274.800  Supplier#000008854
RUSSIA                           76346  Manufacturer#3
1xhLoOUM7I3mZ1mKnerw OSqdbb4QbGa          32-524-148-
5221  y. courts do wake slyly. carefully ironic
platelets haggle above the slyly regular the
          9249.350  Supplier#000003973
FRANCE                           26466  Manufacturer#1
d18GiDsL6Wm2IsGXM,RZf1jCsgZAOjNYVThTRP4   16-722-866-
1658  uests are furiously. regular tithes through the
regular, final accounts cajole furiously above the q
          9249.350  Supplier#000003973
FRANCE                           33972  Manufacturer#1
d18GiDsL6Wm2IsGXM,RZf1jCsgZAOjNYVThTRP4   16-722-866-
1658  uests are furiously. regular tithes through the
regular, final accounts cajole furiously above the q
          9208.700  Supplier#000007769
ROMANIA                          40256  Manufacturer#5
rsimdze 5o9P Ht7xS                        29-964-424-
9649  lites was quickly above the furiously ironic
requests. slyly even foxes against the blithely bold
          9201.470  Supplier#000009690
UNITED KINGDOM                   67183  Manufacturer#5
CB BnUTlmi5zdeEl7R7                       33-121-267-
9529  e even, even foxes. blithely ironic packages
cajole regular packages. slyly final ide
          9192.100  Supplier#000000115
UNITED KINGDOM                   85098  Manufacturer#3
nJ 2t0f7Ve,wL1,6WzGBJLNBUCKlsV            33-597-248-
1220  es across the carefully express accounts boost
caref
          9189.980  Supplier#000001226
GERMANY                          21225  Manufacturer#4
qsLCqSvLyZfuXIpjz                         17-725-903-
1381  deposits. blithely bold excuses about the slyly
bold forges wake
          9128.970  Supplier#000004311
RUSSIA                          146768  Manufacturer#5
I8IjnXd7NSJRs594RxsRR0                    32-155-440-
7120  refully. blithely unusual asymptotes haggle
          9104.830  Supplier#000008520
GERMANY                         150974  Manufacturer#4
```

RqRVDgD0ER J9 b41vR2,3 17-728-804-1793 ly about the blithely ironic depths. slyly final theodolites among the fluffily bold ideas print
9101.000 Supplier#000005791
ROMANIA 128254 Manufacturer#5
zub2zCV,jhHPPQqi,P2INAjE1zI n66cOEoXFG 29-549-251-5384 ts. notornis detect blithely above the carefully bold requests. blithely even package
9094.570 Supplier#000004582
RUSSIA 39575 Manufacturer#1
WB0XkCSG3r,mnQ n,h9VIxjjr9ARHFvKgMDf 32-587-577-1351 jole. regular accounts sleep blithely frets. final pinto beans play furiously past the
8996.870 Supplier#000004702
FRANCE 102191 Manufacturer#5
8XVcQK23akp 16-811-269-8946 ickly final packages along the express plat
8996.140 Supplier#000009814
ROMANIA 139813 Manufacturer#2
af0O5pg83lPU4IDVmEylXZVqYZQzSDlYLAmR 29-995-571-8781 dependencies boost quickly across the furiously pending requests! unusual dolphins play sl
8968.420 Supplier#000010000
ROMANIA 119999 Manufacturer#5
aTGLEusCiL4F PDBdv665XBJhPyCOB0i 29-578-432-2146 ly regular foxes boost slyly. quickly special waters boost carefully ironi
8936.820 Supplier#000007043
UNITED KINGDOM 109512 Manufacturer#1
FVajceZInZdbJE6Z9XsRUxrUEpiwHDrOXi,1Rz 33-784-177-8208 efully regular courts. furiousl
8929.420 Supplier#000008770
FRANCE 173735 Manufacturer#4
R7cG26TtXrHAP9 HckhfRi 16-242-746-9248 cajole furiously unusual requests. quickly stealthy requests are.
8920.590 Supplier#000003967
ROMANIA 26460 Manufacturer#1
eHoAXe62SY9 29-194-731-3944 aters. express, pending instructions sleep. brave, r
8920.590 Supplier#000003967
ROMANIA 173966 Manufacturer#2
eHoAXe62SY9 29-194-731-3944 aters. express, pending instructions sleep. brave, r
8913.960 Supplier#000004603
UNITED KINGDOM 137063 Manufacturer#2
OUzlvMUr7n,utLxmPNeYKSf3T24OXskxB5 33-789-255-7342 haggle slyly above the furiously regular pinto beans. even
8877.820 Supplier#000007967
FRANCE 167966 Manufacturer#5
A3pi1BARM4nx6R,qrwFoRPU 16-442-147-9345 ously foxes. express, ironic requests im
8862.240 Supplier#000003323
ROMANIA 73322 Manufacturer#3
W9 lYcsC9FwBqk3ItL 29-736-951-3710 ly pending ideas sleep about the furiously unu
8841.590 Supplier#000005750
ROMANIA 100729 Manufacturer#5
Erx3lAgu0g62iaHF9x50uMH4EgeN9hEG 29-344-502-5481 gainst the pinto beans. fluffily unusual dependencies affix slyly even deposits.
8781.710 Supplier#000003121
ROMANIA 13120 Manufacturer#5
wNqTogx238ZYCamFb,50v,bj 4IbNFW9Bvw1xP 29-707-291-5144 s wake quickly ironic ideas
8754.240 Supplier#000009407
UNITED KINGDOM 179406 Manufacturer#4
CHRCbkaWcf5B 33-903-970-9604 e ironic requests. carefully even foxes above the furious
8691.060 Supplier#000004429
UNITED KINGDOM 126892 Manufacturer#2
k,BQms5UhoAF1B2Asi,fLib 33-964-337-5038 efully express deposits kindle after the deposits. final
8655.990 Supplier#000006330
RUSSIA 193810 Manufacturer#2
UozlaENr0ytKe2w6CeIEWFWn iO3S8Rae7Ou 32-561-198-3705 symptotes use about the express dolphins. requests use after the express platelets. final, ex
8638.360 Supplier#000002920
RUSSIA 75398 Manufacturer#1
Je2a8bszf3L 32-122-621-7549 ly quickly ironic requests. even requests whithout t
8638.360 Supplier#000002920
RUSSIA 170402 Manufacturer#3
Je2a8bszf3L 32-122-621-7549 ly quickly ironic requests. even requests whithout t
8607.690 Supplier#000006003
UNITED KINGDOM 76002 Manufacturer#2
EH9wADcEiuenM0NR08zDwMidw,52Y2RyILEiA 33-416-807-5206 ar, pending accounts. pending depende
8569.520 Supplier#000005936
RUSSIA 5935 Manufacturer#5
jXaNZ6vwnEWJ2ksLZJpjtgt0bY2a3AU 32-644-251-7916 . regular foxes nag carefully atop the regular, silent deposits. quickly regular packages
8564.120 Supplier#000000033
GERMANY 110032 Manufacturer#1
gfeKpYw3400L0SDywXA6Ya1Qmq1w6YB9f3R 17-138-897-9374 n sauternes along the regular asymptotes are regularly along the
8553.820 Supplier#000003979
ROMANIA 143978 Manufacturer#4
BfmVhCAnCMY3jzpjUMy4CNWs9 HzpdQR7INJU 29-124-646-4897 ic requests wake against the blithely unusual accounts. fluffily r
8517.230 Supplier#000009529
RUSSIA 37025 Manufacturer#5
e44R8o7JAIS9iMcr 32-565-297-8775 ove the even courts. furiously special platelets
8517.230 Supplier#000009529
RUSSIA 59528 Manufacturer#2
e44R8o7JAIS9iMcr 32-565-297-8775 ove the even courts. furiously special platelets
8503.700 Supplier#000006830
RUSSIA 44325 Manufacturer#4
BC4WFCYRUZyaIgchU 4S 32-147-878-5069 pades cajole. furious packages among the carefully express excuses boost furiously across th
8457.090 Supplier#000009456
UNITED KINGDOM 19455 Manufacturer#1
7SBhZs8gP1cJjT0Qf433YBk 33-858-440-4349 cing requests along the furiously unusual deposits promise among the furiously unus
8441.400 Supplier#000003817
FRANCE 141302 Manufacturer#2
hU3fz3xL78 16-339-356-5115 ely even ideas. ideas wake slyly furiously unusual instructions. pinto beans sleep ag
8432.890 Supplier#000003990
RUSSIA 191470 Manufacturer#1
wehBBp1RQbfxAYDASS75MsywmsKHRVdkrvNe6m 32-839-509-9301 ep furiously. packages should have to haggle slyly across the deposits. furiously regu
8431.400 Supplier#000002675
ROMANIA 5174 Manufacturer#1
HJFStOu9R5NGPOegKhgbzBdyvrG2yh8w 29-474-643-1443 ithely express pinto beans. blithely even foxes haggle. furiously regular theodol
8407.040 Supplier#000005406
RUSSIA 162889 Manufacturer#4
j7 gYF5RW8DC5UrjKC 32-626-152-4621 r the blithely regular packages. slyly ironic theodoli
8386.080 Supplier#000008518
FRANCE 36014 Manufacturer#3
2jqzqqAVe9crMVGP,n9nTsQXulNLTUYoJjEDcqWV 16-618-780-7481 blithely bold pains are carefully platelets. finally regular pinto beans sleep carefully special
8376.520 Supplier#000005306
UNITED KINGDOM 190267 Manufacturer#5
9t8Y8 QqSIsoADPt6NLdk,TP5zyRx41oBUlgoGc9 33-632-514-7931 ly final accounts sleep special, regular requests. furiously regular
8348.740 Supplier#000008851
FRANCE 66344 Manufacturer#4
nWxi7GwEbjhw1 16-796-240-2472 boldly final deposits. regular, even instructions detect slyly. fluffily unusual pinto bea
8338.580 Supplier#000007269
FRANCE 17268 Manufacturer#4
ZwhJSwABUoiB04,3 16-267-277-4365 iously final accounts. even pinto beans cajole slyly regular
8328.460 Supplier#000001744
ROMANIA 69237 Manufacturer#5

```
oLo3fV64q2,FKHa3p,qHnS7Yzv,ps8           29-330-728-
5873  ep carefully-- even, careful packages are slyly
along t
          8307.930  Supplier#000003142
GERMANY                        18139  Manufacturer#1
dqblvV8dCNAorGlJ                       17-595-447-
6026  olites wake furiously regular decoys. final
requests nod
          8231.610  Supplier#000009558
RUSSIA                        192000  Manufacturer#2
mcdgen,yT1iJDHDS5fV                    32-762-137-
5858  foxes according to the furi
          8152.610  Supplier#000002731
ROMANIA                        15227  Manufacturer#4
nluXJCuY1tu                            29-805-463-
2030  special requests. even, regular warhorses affix
among the final gr
          8109.090  Supplier#000009186
FRANCE                         99185  Manufacturer#1
wgfosrVPexl9pEXWywaqlBMDYYf            16-668-570-
1402  tions haggle slyly about the sil
          8102.620  Supplier#000003347
UNITED KINGDOM                 18344  Manufacturer#5
m CtXS2S16i                            33-454-274-
8532  egrate with the slyly bold instructions. special
foxes haggle silently among the
          8046.070  Supplier#000008780
FRANCE                        191222  Manufacturer#3
AczzuE0UK9osj ,Lx0Jmh                  16-473-215-
6395  onic platelets cajole after the regular
instructions. permanently bold excuses
          8042.090  Supplier#000003245
RUSSIA                        135705  Manufacturer#4
Dh8Ikg39onrbOL4DyTfGw8a9oKUX3d9Y       32-836-132-
8872  osits. packages cajole slyly. furiously regular
deposits cajole slyly. q
          8042.090  Supplier#000003245
RUSSIA                        150729  Manufacturer#1
Dh8Ikg39onrbOL4DyTfGw8a9oKUX3d9Y       32-836-132-
8872  osits. packages cajole slyly. furiously regular
deposits cajole slyly. q
          7992.400  Supplier#000006108
FRANCE                        118574  Manufacturer#1
8tBydnTDwUqfBfFV4l3                    16-974-998-
8937  ironic ideas? fluffily even instructions wake.
blithel
          7980.650  Supplier#000001288
FRANCE                         13784  Manufacturer#4
zE,7HgVPrCn                            16-646-464-
8247  ully bold courts. escapades nag slyly. furiously
fluffy theodo
          7950.370  Supplier#000008101
GERMANY                        33094  Manufacturer#5
kkYvL6IuvojJgTNG IKkaXQDYgx8ILohj      17-627-663-
8014  arefully unusual requests x-ray above the
quickly final deposits.
          7937.930  Supplier#000009012
ROMANIA                        83995  Manufacturer#2
iUiTziH,Ek3i4lwSgunXMgrcTzwdb          29-250-925-
9690  to the blithely ironic deposits nag sly
          7914.450  Supplier#000001013
RUSSIA                        125988  Manufacturer#2
riRcntps4KEDtYScjpMIWeYF6mNnR          32-194-698-
3365  busily bold packages are dolphi
          7912.910  Supplier#000004211
GERMANY                       159180  Manufacturer#5
2wQRVovHrm3,v03IKzfTd,1PYsFXQFFOG      17-266-947-
7315  ay furiously regular platelets. cou
          7912.910  Supplier#000004211
GERMANY                       184210  Manufacturer#4
2wQRVovHrm3,v03IKzfTd,1PYsFXQFFOG      17-266-947-
7315  ay furiously regular platelets. cou
          7894.560  Supplier#000007981
GERMANY                        85472  Manufacturer#4
NSJ96vMROAbeXP                         17-963-404-
3760  ic platelets affix after the furiously
          7887.080  Supplier#000009792
GERMANY                       164759  Manufacturer#3
Y28ITVeYriT3kIGdV2K8fSZ V2UqT5H1Otz    17-988-938-
4296  ckly around the carefully fluffy theodolites.
slyly ironic pack
          7871.500  Supplier#000007206
RUSSIA                        104695  Manufacturer#1
3w fNCnrVmvJjE95sgWZzvW                32-432-452-
7731  ironic requests. furiously final theodolites
cajole. final, express packages sleep. quickly reg
          7852.450  Supplier#000005864
RUSSIA                         8363  Manufacturer#4
WCNfBPZeSXh3h,c                        32-454-883-
3821  usly unusual pinto beans. brave ideas sleep
carefully quickly ironi
          7850.660  Supplier#000001518
UNITED KINGDOM                 86501  Manufacturer#1
ONda3YJiHKJOC                          33-730-383-
3892  ifts haggle fluffily pending pai
          7843.520  Supplier#000006683
FRANCE                         11680  Manufacturer#4
2Z0JGkiv01Y00oCFwUGfviIbhzCdy          16-464-517-
8943  express, final pinto beans x-ray slyly
asymptotes. unusual, unusual


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


Stop timestamp 09/28/07 10:30:15.972013
Query Time =             0.5 secs

==================================================
```

qryqual03

```
==================================================
Start timestamp 09/28/07 10:34:57.541872

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

-- 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 09/28/07 10:35:08.899802
Query Time =            11.4 secs
==================================================
```

qryqual04

```
==================================================
Start timestamp 09/28/07 10:35:10.286130

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

-- 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 09/28/07 10:36:12.943880
Query Time =            62.7 secs
==================================================
```

qryqual05

```
==================================================
Start timestamp 09/28/07 10:38:29.270426

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

-- 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 09/28/07 10:39:33.714519
Query Time =            64.4 secs
==================================================
```

qryqual06

```
==================================================
Start timestamp 09/28/07 10:31:34.711041

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

-- 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 09/28/07 10:31:35.036127
Query Time =            0.3 secs

==================================================
```

qryqual07

```
==================================================
Start timestamp 09/28/07 10:39:33.714519

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

-- 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 09/28/07 10:39:37.747708
Query Time =             4.0 secs
=====================================================
```

qryqual08

```
=====================================================
Start timestamp 09/28/07 10:32:42.321508

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

-- 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 09/28/07 10:33:47.643079
Query Time =            65.3 secs
=====================================================
```

qryqual09

```
=====================================================

Start timestamp 09/28/07 10:30:15.972013

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

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

Tag: Q9     Stream: -1   Sequence number: 3

select
nation,
o_year,
sum(amount) as sum_profit
from
(
select
n_name as nation,
year(o_orderdate) as o_year,
l_extendedprice * (1 - l_discount) - ps_supplycost *
l_quantity as amount
from
tpcd.part,
tpcd.supplier,
tpcd.lineitem,
tpcd.partsupp,
tpcd.orders,
tpcd.nation
where
s_suppkey = l_suppkey
and ps_suppkey = l_suppkey
and ps_partkey = l_partkey
and p_partkey = l_partkey
and o_orderkey = l_orderkey
and s_nationkey = n_nationkey
and p_name like '%green%'
) as profit
group by
nation,
o_year
order by
nation,
o_year desc

NATION                     O_YEAR       SUM_PROFIT
-----------------------------------------------------
--------
ALGERIA                           1998
31342867.235
ALGERIA                           1997
57138193.023
ALGERIA                           1996
56140140.133
ALGERIA                           1995
53051469.653
ALGERIA                           1994
53867582.129
ALGERIA                           1993
```

54942718.132
ALGERIA 1992
54628034.713
ARGENTINA 1998
30211185.708
ARGENTINA 1997
50805741.752
ARGENTINA 1996
51923746.576
ARGENTINA 1995
49298625.767
ARGENTINA 1994
50835610.109
ARGENTINA 1993
51646079.178
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
30379833.768
CANADA 1997
50465052.311
CANADA 1996
52560501.390
CANADA 1995
52375332.809
CANADA 1994
52600364.659
CANADA 1993
52644504.074
CANADA 1992
53932871.697
CHINA 1998
31075466.165
CHINA 1997
50551874.450
CHINA 1996
51039293.875
CHINA 1995
49287534.617
CHINA 1994
50851090.067
CHINA 1993
54229629.833
CHINA 1992
52400529.372
EGYPT 1998
29054433.386
EGYPT 1997
50627611.452
EGYPT 1996
49542212.845
EGYPT 1995
48311550.321
EGYPT 1994
49790644.736
EGYPT 1993
48904292.969
EGYPT 1992
49434932.619
ETHIOPIA 1998
28040717.267
ETHIOPIA 1997
47455009.866
ETHIOPIA 1996
46491097.573
ETHIOPIA 1995
46804449.301
ETHIOPIA 1994
48516143.917
ETHIOPIA 1993
46551891.563
ETHIOPIA 1992
44934648.643
FRANCE 1998
32226407.839
FRANCE 1997
47121485.860
FRANCE 1996
47263135.496
FRANCE 1995
47275997.571
FRANCE 1994
47067209.331
FRANCE 1993
51163370.106
FRANCE 1992
47846235.331
GERMANY 1998
28624942.660
GERMANY 1997
49309074.877
GERMANY 1996
49918683.168
GERMANY 1995
52650718.724
GERMANY 1994
50346900.423
GERMANY 1993
50991895.806
GERMANY 1992
48274126.099
INDIA 1998
29943144.354
INDIA 1997
50665453.231
INDIA 1996
50283092.291
INDIA 1995
50006774.645
INDIA 1994
48995190.756
INDIA 1993
50286902.853
INDIA 1992
50850329.402
INDONESIA 1998
27672339.997
INDONESIA 1997
50512145.726
INDONESIA 1996
51653060.117
INDONESIA 1995
51508779.594
INDONESIA 1994
52817950.322
INDONESIA 1993
47959994.955
INDONESIA 1992
51776605.032
IRAN 1998
29065736.238
IRAN 1997
50042063.054
IRAN 1996
50926653.188
IRAN 1995
51249667.649
IRAN 1994
50337085.865
IRAN 1993
51730763.490
IRAN 1992
49955856.563
IRAQ 1998
31624551.002
IRAQ 1997
55121749.019
IRAQ 1996
55897663.794
IRAQ 1995
54815472.517
IRAQ 1994
54408516.127
IRAQ 1993
53633167.977
IRAQ 1992
55891939.340
JAPAN 1998

```
27934179.669
JAPAN                          1997
44517162.546
JAPAN                          1996
42545606.120
JAPAN                          1995
43749356.400
JAPAN                          1994
44840243.070
JAPAN                          1993
44660015.533
JAPAN                          1992
45410249.122
JORDAN                         1998
26901488.578
JORDAN                         1997
45471878.410
JORDAN                         1996
46794325.792
JORDAN                         1995
45178828.576
JORDAN                         1994
45333636.508
JORDAN                         1993
47971496.098
JORDAN                         1992
44717239.177
KENYA                          1998
28597614.337
KENYA                          1997
47949733.727
KENYA                          1996
46886924.623
KENYA                          1995
46072338.755
KENYA                          1994
45772061.171
KENYA                          1993
46308728.235
KENYA                          1992
47257780.841
MOROCCO                        1998
26732115.580
MOROCCO                        1997
45637304.250
MOROCCO                        1996
45558221.745
MOROCCO                        1995
47851318.887
MOROCCO                        1994
46272172.945
MOROCCO                        1993
46764326.182
MOROCCO                        1992
48122783.583
MOZAMBIQUE                     1998
30712392.011
MOZAMBIQUE                     1997
50316528.762
MOZAMBIQUE                     1996
51640320.251
MOZAMBIQUE                     1995
50693774.506
MOZAMBIQUE                     1994
49253277.626
MOZAMBIQUE                     1993
49153016.537
MOZAMBIQUE                     1992
48247551.850
PERU                           1998
29326102.320
PERU                           1997
49753780.395
PERU                           1996
50935170.293
PERU                           1995
53309883.407
PERU                           1994
50643531.797
PERU                           1993
51584622.002
PERU                           1992
47523899.055
ROMANIA                        1998
30368667.400
ROMANIA                        1997
50365683.853
ROMANIA                        1996
49598999.015
ROMANIA                        1995
47537642.870
ROMANIA                        1994
51455283.010
ROMANIA                        1993
50407136.892
ROMANIA                        1992
48185385.129
RUSSIA                         1998
28322384.027
RUSSIA                         1997
50106685.182
RUSSIA                         1996
51753342.430
RUSSIA                         1995
49215820.365
RUSSIA                         1994
52205666.441
RUSSIA                         1993
51860230.034
RUSSIA                         1992
53251677.153
SAUDI ARABIA                   1998
31541259.810
SAUDI ARABIA                   1997
52438750.808
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
48086499.712
UNITED KINGDOM                 1993
49166827.223
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
49658284.613
VIETNAM                        1994
50596057.261
VIETNAM                        1993
50953919.152
VIETNAM                        1992
49613838.315

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

```
Stop timestamp 09/28/07 10:31:32.538993
Query Time =            76.6 secs

================================================
```

qryqual10

```
================================================
Start timestamp 09/28/07 10:36:24.768082

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

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

Tag: Q10    Stream: -1   Sequence number: 18

select
c_custkey,
c_name,
sum(l_extendedprice * (1 - l_discount)) as revenue,
c_acctbal,
n_name,
c_address,
c_phone,
c_comment
from
tpcd.customer,
tpcd.orders,
tpcd.lineitem,
tpcd.nation
where
c_custkey = o_custkey
and l_orderkey = o_orderkey
and o_orderdate >= date ('1993-10-01')
and o_orderdate < date ('1993-10-01') + 3 month
and l_returnflag = 'R'
and c_nationkey = n_nationkey
group by
c_custkey,
c_name,
c_acctbal,
c_phone,
n_name,
c_address,
c_comment
order by
revenue desc
fetch first 20 rows only

C_CUSTKEY    C_NAME                         REVENUE
C_ACCTBAL            N_NAME
C_ADDRESS                                     C_PHONE
C_COMMENT
-------------------------------------------------------
-------------------------------------------------------
-------------------------------------------------------
-------------------------------------------------------
-------------------------------------------------------
------------------
      57040  Customer#000057040
734235.246               632.870  JAPAN
Eioyzjf4pp                                    22-895-641-
3466  sits. slyly regular requests sleep alongside of
the regular inst
     143347  Customer#000143347
721002.695              2557.470  EGYPT
1aReFYv,Kw4                                   14-742-935-
3718  ggle carefully enticing requests. final deposits
use bold, bold pinto beans. ironic, idle re
      60838  Customer#000060838
679127.308              2454.770  BRAZIL
64EaJ5vMAHWJlBOxJklpNc2RJiWE                  12-913-494-
9813  need to boost against the slyly regular account
     101998  Customer#000101998
637029.567              3790.890  UNITED KINGDOM
01c9CILnNtfOQYmZj                             33-593-865-
6378  ress foxes wake slyly after the bold excuses.
ironic platelets are furiously carefully bold
theodolites
     125341  Customer#000125341
633508.086              4983.510  GERMANY
S29ODD6bceU8QSuuEJznkNaK                      17-582-695-
5962  arefully even depths. blithely even excuses
sleep furiously. foxes use except the dependencies. ca
      25501  Customer#000025501
620269.785              7725.040  ETHIOPIA
W556MXuoiaYCCZamJI,Rn0B4ACUGdkQ8DZ            15-874-808-
6793  he pending instructions wake carefully at the
pinto beans. regular, final instructions along the
slyly fina
     115831  Customer#000115831
596423.867              5098.100  FRANCE
rFeBbEEyk dl ne7zV5fDrmiq1oK09wV7pxqCgIc  16-715-386-
3788  l somas sleep. furiously final deposits wake
blithely regular pinto b
      84223  Customer#000084223
594998.024               528.650  UNITED KINGDOM
nAVZCs6BaWap rrM27N 2qBnzc5WBauxbA             33-442-824-
8191  slyly final deposits haggle regular, pending
dependencies. pending escapades wake
      54289  Customer#000054289
585603.392              5583.020  IRAN
vXCxoCsU0Bad5JQI ,oobkZ                       20-834-292-
4707  ely special foxes are quickly finally ironic p
      39922  Customer#000039922
584878.113              7321.110  GERMANY
Zgy4s50l2GKN4pLDPBU8m342gIw6R                 17-147-757-
8036  y final requests. furiously final foxes cajole
blithely special platelets. f
       6226  Customer#000006226
576783.761              2230.090  UNITED KINGDOM
8gPu8,NPGkfyQQ0hcIYUGPIBWc,ybP5g,             33-657-701-
3391  ending platelets along the express deposits
cajole carefully final
        922  Customer#000000922
576767.533              3869.250  GERMANY
Az9RFaut7NkPnc5zSD2PwHgVwr4jRzq               17-945-916-
9648  luffily fluffy deposits. packages c
     147946  Customer#000147946
576455.132              2030.130  ALGERIA
iANyZHjqhyy7Ajah0pTrYyhJ                      10-886-956-
3143  ithely ironic deposits haggle blithely ironic
requests. quickly regu
     115640  Customer#000115640
569341.193              6436.100  ARGENTINA
Vtgfia9qI 7EpHgecU1X                          11-411-543-
4901  ost slyly along the patterns; pinto be
      73606  Customer#000073606
568656.858              1785.670  JAPAN
xuR0Tro5yChDfOCrjkd2ol                        22-437-653-
6966  he furiously regular ideas. slowly
     110246  Customer#000110246
566842.981              7763.350  VIETNAM
7KzflgX MDOq7sOkI                             31-943-426-
9837  egular deposits serve blithely above the fl
     142549  Customer#000142549
563537.237              5085.990  INDONESIA
ChqEoK43OysjdHbtKCp6dKqjNyvvi9                19-955-562-
2398  sleep pending courts. ironic deposits against
the carefully unusual platelets cajole carefully
express accounts.
     146149  Customer#000146149
557254.986              1791.550  ROMANIA
s87fvzFQpU                                    29-744-164-
6487  of the slyly silent accounts. quickly final
accounts across the
      52528  Customer#000052528
556397.351               551.790  ARGENTINA
NFztyTOR10UOJ                                 11-208-192-
3205  deposits hinder. blithely pending asymptotes
breach slyly regular re
      23431  Customer#000023431
554269.536              3381.860  ROMANIA
HgiV0phqhaIa9aydNoIlb                         29-915-458-
2654  nusual, even instructions: furiously stealthy n


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


Stop timestamp 09/28/07 10:37:27.110036
Query Time =            62.3 secs

================================================
```

qryqual11

```
================================================
Start timestamp 09/28/07 10:36:12.943880

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

```
-- 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
      26489          13377256.380
      96493          13339057.830
      56548          13329014.970
      55576          13306843.350
     159751          13306614.480
      92406          13287414.500
     182636          13223726.740
     199969          13135288.210
      62865          13001926.940
       7284          12945298.190
     197867          12944510.520
      11562          12931575.510
      75165          12916918.120
      97175          12911283.500
     140840          12896562.230
      65241          12890600.460
     166120          12876927.220
       9035          12863828.700
     144616          12853549.300
     176723          12832309.740
     170884          12792136.580
      29790          12723300.330
      95213          12555483.730
     183873          12550533.050
     171235          12476538.300
      21533          12437821.320
      17290          12432159.500
     156397          12260623.500
     122611          12222812.980
     139155          12220319.250
     146316          12215800.610
     171381          12199734.520
     198633          12078226.950
     167417          12046637.620
      59512          12043468.760
      31688          12034893.640
     159586          12001505.840
       8993          11963814.300
     120302          11857707.550
      43536          11779340.520
       9552          11776909.160
      86223          11772205.080
      53776          11758669.650
     131285          11616953.740
      91628          11611114.830
     169644          11567959.720
     182299          11567462.050
      33107          11453818.760
     104184          11436657.440
      67027          11419127.140
     176869          11371451.710
      30885          11369674.790
      54420          11345076.880
      72240          11313951.050
     178708          11294635.170
      81298          11273686.130
     158324          11243442.720
     117095          11242535.240
     176793          11237733.380
      86091          11177793.790
     116033          11145434.360
     129058          11119112.200
     193714          11104706.390
     117195          11077217.960
      49851          11043701.780
      19791          11030662.620
      75800          11012401.620
     161562          10996371.690
      10119          10980015.750
      39185          10970042.560
      47223          10950022.130
     175594          10942923.050
     111295          10893675.610
     155446          10852764.570
     156391          10839810.380
      40884          10837234.190
      83532           8009753.850
----- lines deleted ----
      50166           8007137.890
     181562           8006805.960
     175165           8005319.760
      62500           8005316.280
      36342           8004333.400
     128435           8004242.880
      92516           8003836.800
      30802           8003710.880
     107418           8000430.300
      46620           7999778.350
     191803           7994734.150
     106343           7993087.760
      59362           7990397.460
       8329           7990052.900
      75133           7988244.000
     179023           7986829.620
     135899           7985726.640
       5824           7985340.020
     148579           7984889.560
      95888           7984735.720
       9791           7982699.790
     170437           7982370.720
      39782           7977858.240
      20605           7977556.000
      28682           7976960.000
      42172           7973399.000
      56137           7971405.400
      64729           7970769.720
      98643           7968603.730
     153787           7967535.580
       8932           7967222.190
      20134           7965713.280
```

```
197635          7963507.580
 80408          7963312.170
 37728          7961875.680
 26624          7961772.310
 44736          7961144.100
 29763          7960605.030
 36147          7959463.680
146040          7957587.660
115469          7957485.140
142276          7956790.630
181280          7954037.350
115096          7953047.550
109650          7952258.730
 93862          7951992.240
158325          7950728.300
 55952          7950387.060
122397          7947106.270
 28114          7946945.720
 11966          7945197.480
 47814          7944083.000
 85096          7943691.060
 51657          7943593.770
196680          7943578.890
 13141          7942730.340
193327          7941036.250
152612          7940663.710
139680          7939242.360
 31134          7938318.300
 45636          7937240.850
 56694          7936015.950
  8114          7933921.880
 71518          7932261.690
 72922          7930400.640
146699          7929167.400
 92387          7928972.670
186289          7928786.190
 95952          7927972.780
196514          7927180.700
  4403          7925729.040
  2267          7925649.370
 45924          7925047.680
 11493          7916722.230
104478          7916253.600
166794          7913842.000
161995          7910874.270
 23538          7909752.060
 41093          7909579.920
112073          7908617.570
 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 09/28/07 10:36:13.355787
Query Time =             0.4 secs

==================================================
```

qryqual12

```
Start timestamp 09/28/07 10:39:37.747708

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

-- 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 09/28/07 10:39:40.791631
Query Time =             3.0 secs
==================================================
```

qryqual13

```
==================================================
Start timestamp 09/28/07 10:34:55.250082

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

-- 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	50005
9	6641
10	6532
11	6014
8	5937
12	5639
13	5024
19	4793
7	4687
17	4587
18	4529
20	4516
15	4505
14	4446
16	4273
21	4190
22	3623
6	3265
23	3225
24	2742
25	2086
5	1948
26	1612
27	1179
4	1007
28	893
29	593
3	415
30	376
31	226
32	148
2	134
33	75
34	50
35	37
1	17
36	14
38	5
37	5
40	4
41	2
39	1

Number of rows retrieved is: 42

Stop timestamp 09/28/07 10:34:57.541872
Query Time = 2.3 secs

===

qryqual14

===
Start timestamp 09/28/07 10:30:14.770581

--#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 09/28/07 10:30:15.445861
Query Time = 0.7 secs
===

qryqual15

===
Start timestamp 09/28/07 10:36:13.355787

-- 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 09/28/07 10:36:14.073047
Query Time = 0.7 secs

===

qryqual16

===
Start timestamp 09/28/07 10:35:09.428342

-- 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
Brand#14    PROMO ANODIZED NICKEL               45
23
Brand#32    ECONOMY PLATED BRASS                 9
23
Brand#52    SMALL ANODIZED COPPER                3
23
Brand#11    ECONOMY BRUSHED COPPER              45
20
Brand#11    ECONOMY PLATED BRASS                23
20
Brand#11    LARGE BRUSHED COPPER                49
20
Brand#11    LARGE POLISHED COPPER               49
20
Brand#12    STANDARD ANODIZED TIN               49
20
Brand#12    STANDARD PLATED BRASS               19
20
Brand#13    ECONOMY BRUSHED BRASS                9
20
Brand#13    ECONOMY BURNISHED STEEL             14
20
Brand#13    LARGE BURNISHED NICKEL              19
20
Brand#13    MEDIUM BURNISHED COPPER             36
20
Brand#13    SMALL BRUSHED TIN                   45
20
Brand#13    STANDARD ANODIZED COPPER             3
20
Brand#13    STANDARD PLATED NICKEL              23
20
Brand#14    ECONOMY ANODIZED COPPER             14
20
Brand#14    ECONOMY PLATED TIN                  36
20
Brand#14    ECONOMY POLISHED NICKEL              3
20
Brand#14    MEDIUM ANODIZED NICKEL               3
20
Brand#14    SMALL POLISHED TIN                  14
20
Brand#15    MEDIUM ANODIZED COPPER               9
20
Brand#15    MEDIUM PLATED TIN                   23
20
Brand#15    PROMO PLATED BRASS                  14
20
Brand#15    SMALL ANODIZED COPPER               45
20
---- lines deleted ----
Brand#55    STANDARD BURNISHED STEEL            36
4
Brand#55    STANDARD BURNISHED STEEL            45
4
Brand#55    STANDARD BURNISHED TIN               9
4
Brand#55    STANDARD BURNISHED TIN              19
4
Brand#55    STANDARD BURNISHED TIN              36
4
Brand#55    STANDARD BURNISHED TIN              49
4
Brand#55    STANDARD PLATED BRASS                9
4
Brand#55    STANDARD PLATED BRASS               45
4
Brand#55    STANDARD PLATED BRASS               49
4
Brand#55    STANDARD PLATED COPPER               9
4
Brand#55    STANDARD PLATED COPPER              45
4
Brand#55    STANDARD PLATED NICKEL               3
4
Brand#55    STANDARD PLATED NICKEL              19
4
Brand#55    STANDARD PLATED NICKEL              45
4
Brand#55    STANDARD PLATED STEEL               14
4
Brand#55    STANDARD PLATED STEEL               23
4
```

```
Brand#55    STANDARD PLATED STEEL               49
4
Brand#55    STANDARD PLATED TIN                  9
4
Brand#55    STANDARD PLATED TIN                 14
4
Brand#55    STANDARD PLATED TIN                 36
4
Brand#55    STANDARD POLISHED BRASS              3
4
Brand#55    STANDARD POLISHED BRASS              9
4
Brand#55    STANDARD POLISHED BRASS             23
4
Brand#55    STANDARD POLISHED COPPER             3
4
Brand#55    STANDARD POLISHED COPPER            23
4
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 09/28/07 10:35:10.286130
Query Time =             0.9 secs

==================================================
```

qryqual17

```
==================================================
Start timestamp 09/28/07 10:31:35.036127

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

-- 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 09/28/07 10:32:37.154803
Query Time =          62.1 secs

==================================================
```

qryqual18

```
==================================================
Start timestamp 09/28/07 10:32:37.154803

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

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

Tag: Q18    Stream: -1   Sequence number: 7

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

```
C_NAME                      C_CUSTKEY    O_ORDERKEY
O_ORDERDATE  O_TOTALPRICE          6
-------------------------------------------------------
-------------------------------------------------------
----------
Customer#000128120              128120
4722021  1994-04-07             544089.090
323.000
Customer#000144617              144617
3043270  1997-02-12             530604.440
317.000
Customer#000013940               13940
2232932  1997-04-13             522720.610
304.000
Customer#000066790               66790
2199712  1996-09-30             515531.820
327.000
Customer#000046435               46435
4745607  1997-07-03             508047.990
309.000
Customer#000015272               15272
3883783  1993-07-28             500241.330
302.000
Customer#000146608              146608
3342468  1994-06-12             499794.580
303.000
Customer#000096103               96103
5984582  1992-03-16             494398.790
312.000
Customer#000024341               24341
1474818  1992-11-15             491348.260
302.000
Customer#000137446              137446
5489475  1997-05-23             487763.250
311.000
Customer#000107590              107590
4267751  1994-11-04             485141.380
301.000
Customer#000050008               50008
2366755  1996-12-09             483891.260
302.000
Customer#000015619               15619
3767271  1996-08-07             480083.960
318.000
Customer#000077260               77260
1436544  1992-09-12             479499.430
307.000
Customer#000109379              109379
5746311  1996-10-10             478064.110
302.000
Customer#000054602               54602
5832321  1997-02-09             471220.080
307.000
Customer#000105995              105995
2096705  1994-07-03             469692.580
307.000
Customer#000148885              148885
2942469  1992-05-31             469630.440
313.000
Customer#000114586              114586
551136  1993-05-19              469605.590
308.000
Customer#000105260              105260
5296167  1996-09-06             469360.570
303.000
Customer#000147197              147197
1263015  1997-02-02             467149.670
320.000
Customer#000064483               64483
2745894  1996-07-04             466991.350
304.000
Customer#000136573              136573
2761378  1996-05-31             461282.730
301.000
Customer#000016384               16384
502886  1994-04-12              458378.920
312.000
Customer#000117919              117919
2869152  1996-06-20             456815.920
317.000
Customer#000012251               12251
735366  1993-11-24              455107.260
309.000
Customer#000120098              120098
1971680  1995-06-14             453451.230
308.000
Customer#000066098               66098
5007490  1992-08-07             453436.160
304.000
Customer#000117076              117076
4290656  1997-02-05             449545.850
301.000
Customer#000129379              129379
4720454  1997-06-07             448665.790
303.000
Customer#000126865              126865
4702759  1994-11-07             447606.650
320.000
Customer#000088876               88876
983201  1993-12-30              446717.460
304.000
Customer#000036619               36619
4806726  1995-01-17             446704.090
328.000
Customer#000141823              141823
2806245  1996-12-29             446269.120
310.000
Customer#000053029               53029
2662214  1993-08-13             446144.490
302.000
Customer#000018188               18188
3037414  1995-01-25             443807.220
308.000
Customer#000066533               66533
29158  1995-10-21               443576.500
305.000
Customer#000037729               37729
4134341  1995-06-29             441082.970
309.000
Customer#000003566                3566
2329187  1998-01-04             439803.360
304.000
Customer#000045538               45538
4527553  1994-05-22             436275.310
305.000
Customer#000081581               81581
4739650  1995-11-04             435405.900
305.000
Customer#000119989              119989
1544643  1997-09-20             434568.250
320.000
Customer#000003680                3680
3861123  1998-07-03             433525.970
301.000
Customer#000113131              113131
967334  1995-12-15              432957.750
301.000
Customer#000141098              141098
565574  1995-09-24              430986.690
301.000
Customer#000093392               93392
5200102  1997-01-22             425487.510
304.000
Customer#000015631               15631
1845057  1994-05-12             419879.590
302.000
Customer#000112987              112987
4439686  1996-09-17             418161.490
305.000
Customer#000012599               12599
4259524  1998-02-12             415200.610
304.000
Customer#000105410              105410
4478371  1996-03-05             412754.510
302.000
Customer#000149842              149842
5156581  1994-05-30             411329.350
302.000
Customer#000010129               10129
5849444  1994-03-21             409129.850
309.000
Customer#000069904               69904
1742403  1996-10-19             408513.000
305.000
Customer#000017746               17746
6882  1997-04-09                408446.930
303.000
Customer#000013072               13072
1481925  1998-03-15             399195.470
301.000
Customer#000082441               82441
```

```
857959  1994-02-07              382579.740
305.000
Customer#000088703                88703
2995076  1994-01-30              363812.120
302.000


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


Stop timestamp 09/28/07 10:32:42.321508
Query Time =             5.2 secs


==================================================
```

qryqual19

```
==================================================
Start timestamp 09/28/07 10:37:27.110036

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

-- 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 09/28/07 10:38:29.270426
Query Time =          62.2 secs

==================================================
```

qryqual20

```
==================================================
Start timestamp 09/28/07 10:31:32.538993

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

-- 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,tBpNwKb5xMUzeohxlRn,
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
Supplier#000001726 TeRY7TtTH24sEword7yAaSkjx8
Supplier#000001730 Rc8e,1Pybn r6zo0VJIEiD0UD
vhk
Supplier#000001746
qWsendlOekQG1aW4uq06uQaCm51se8lirv7 hBRd
Supplier#000001752 Fra7outx41THYJaRThdOGiBk
Supplier#000001856
jXcRgzYF0ah05iR8p6w5SbJJLcUGyYiURPvFwUWM
Supplier#000001931 FpJbMU2h6ZR2eBv8I9NIxF
Supplier#000001939 Nrk,JA4bfReUs
Supplier#000001990
DSDJkCgBJzuPg1yuM,CUdLnsRliOxkkHezTCA
Supplier#000002020 jB6r1d7MxP6co
Supplier#000002022 dwebGX7Id2pc25YvY33
Supplier#000002036 20ytTtVObjKUUI2WCB0A
Supplier#000002204
uYmlr46C06udCqanj0KiRsoTQakZsEyssL
Supplier#000002243 nSOEV3JeOU79
Supplier#000002245
hz2qWXWVjOyKhqPYMoEwz6zFkrTaDM
Supplier#000002282
ES21K9dxoW1I1TzWCj7ekdlNwSWnv1Z 6mQ,BKn
Supplier#000002303 nCoWfpB6YOymbgOht7ltfklpkHl
Supplier#000002373 RzHSxOTQmElCjxIBiVA52Z
JB58rJhPRylR
Supplier#000002419 qydBQd14I5l5mVXa4fYY
Supplier#000002481
nLKHUOn2Ml9TOA06Znq9GEMcIlMO2
Supplier#000002571 JZUugz04c iJFLrlGsz9O N,W
1rVHNIReyq
Supplier#000002585
CsPoKpw2QuTY4AV1NkWuttneIa4SN
Supplier#000002630 ZIQAvjNUY9KH5ive zm7k
VlPiDl7CCo21
Supplier#000002719
4nnzQI2CbqREQUuIsXTBVUkaP4mNS3
Supplier#000002721 HVdFAN2JHMQSpKm
Supplier#000002730 lIFxR4fzm31C6,muzJwl84z
Supplier#000002775 yDclaDaBD4ihH
Supplier#000002853 rTNAOItXka
Supplier#000002875 6JgMi
9Qt6VmwL3Ltt1SRlKww0keLQ,RAZa
Supplier#000002934 m,trBENywSArwg3DhB
Supplier#000002941 Naddba 8YTEKekZyP0
Supplier#000002960
KCPCEsRGGo6vx8TygHh60nAYf9rStQT2T
Supplier#000002980
B9k9yVsyaXvWktOSHezqHiAEp9id0SKzkw
Supplier#000003062 LSQNgqY1xnOzz9zBCapy7HwOZQ
Supplier#000003087 ANwe8QsZ4rgj1HSqVz991eWQ
Supplier#000003089 s5b VCIZqMSZVa r
g7LTdcg29GbTE7rI1x
Supplier#000003095 HxON3jJhUi3zjt,r mTD
Supplier#000003201
E87yws6I,t0qNs4QW7UzExKiJnJDZWue
Supplier#000003213 pxrRP4irQ1VoyfQ,dTf3
Supplier#000003241 j06SU,LS9O3mwjAMOViANeIhb
Supplier#000003275 9xO4nyJ2QJcX6vGf
Supplier#000003288
EDdfNt7E5Uc,xLTupoIgYL4yY7ujh,
Supplier#000003313
El2I7we,049SPrvomUm4hZwJoOhZkvLxLJXgVH
Supplier#000003314
jnisU8MzqO4iUB3zsPcrysMw3DDUojS4q7LD
Supplier#000003380
jPv0V,pszouuFT3YsAqlP,kxT3u,gTFiEbRt,x
Supplier#000003403 e3X2o ,KCG9tsHji8A
XXCxiF2hZWBw
Supplier#000003421 Sh3dt9W5oeofFWovnFhrg,
Supplier#000003441 zvFJIzS,oUuShHjpcX
Supplier#000003590 sy79CMLxqb,Cbo
Supplier#000003607 lNqFHQYjwSAkf
Supplier#000003625
qY588W0Yk5iaUy1RXTgNrEKrMAjBYHcKs
Supplier#000003656 eEYmmO2gmD
JdfG32XtDgJV,db56
Supplier#000003782
iVsPZg7bk06TqNMwi0LKbLUrC1zmrg
Supplier#000003918 meRvRCsJoAbfqd0Re4
Supplier#000003941 Pmb05mQfBMS618O7WKqZJ 9vyv
Supplier#000003994 W00LZp3NjK0
Supplier#000004005 V723F1wCy2eA4OgIu8TjBtOVUHp
Supplier#000004033 ncsAhv9Je,kFXTNjfb2
Supplier#000004140 0hL7DJyYjcHL
Supplier#000004165
wTJ2dZNQA8P2oi99N6DT47ndHy,XKD2
Supplier#000004207
tF64pwiOM4IkWjN3mS,e06WuAjLx
Supplier#000004236
dl,HPtJmGipxYsSqn9wmqkuWjst,mCeJ8O6T
Supplier#000004246 Xha aXQF7u4qU3LsHD
Supplier#000004278 bBddbpBxIVp Di9
Supplier#000004343 GK3sbopqrQEkWLMvVBFCG
Supplier#000004346 S3076LEOwo
Supplier#000004388 VfZ l1J,mwp4aS
Supplier#000004406
Ah0ZaLu6VwufPWUz,7kbXgYZhauEaHqGIg
Supplier#000004430
yvSsKNSTL5HLXBET4luOsPNLxKzAMk
Supplier#000004522
xXtCKwsZDArxIBGDfzX2PgobGZsBg
Supplier#000004527 p pVXCnxgcklWF6A1o3OHY3qW6
Supplier#000004542 NJSbLJDroYG2y1r3rDiKg
Supplier#000004574 1HvGwnVueZ5CIndc
Supplier#000004655 67NqBc4 t3PG3F8aO
IsqWNq4kGaPowYL
Supplier#000004701 6jX4u47URzIMHf
Supplier#000004711 bEzjp1QdQu ls2ERMxv0km
vn6bu2zXlL1
Supplier#000004987 UFx1upJ8MvOvgFjA8
Supplier#000005000 DeX804 w0H8FrCUvahgy
ilbuzBX3NK
Supplier#000005100 OfvYPs3Io,wEvvLHNaLuCX
Supplier#000005192 JDp4rhXiDw0kf6RH
Supplier#000005195 Woi3b2ZaicPh ZSfu1EfXhE
Supplier#000005283 5fxYXxwXy,TQX,MqDC2hxzyQ
Supplier#000005300 gXG28YqpxU
Supplier#000005386 Ub6AAfHpWLWP
Supplier#000005426 9Dz2OVT1q
sb4BK71ljQ1XjPBYRPvO
Supplier#000005484 saFdOR
qW7AFY,3asPqiiAa11Mo22pCoN0BtPrKo
Supplier#000005505 d2sbjG43KwMPX
Supplier#000005506 On f5ypzoWgB
Supplier#000005516
XsN99Ks9wEvcohU6jRD2MeebQFf76mD8vovuY
Supplier#000005536
Nzo9tGkpgbHT,EZ4D,77MYKl4ah1C
Supplier#000005605 7Vj6Eil0mThqkM
Supplier#000005631
14TVrjlzo2SJEBYCDgpMwTlvwSqC
Supplier#000005730 5rkb0PSews
HvxkL8JaD41UpnSF2cg8H1
Supplier#000005736 2dq XTYhtYWSfp
Supplier#000005737 dmEWcS32C3kx,d,B95 OmYn48
Supplier#000005797 ,o,OebwRbSDmVl9gN9fpWPCiqB
UogvlSR
Supplier#000005836 tx3SjPD2ZuWGFBRH,
Supplier#000005875
lK,sYiGzB94hSyHy9xvSZFbVQNCZe2LXZuGbS
Supplier#000005974
REhR5jE,lLusQXvf54SwYySgsSSVFhu
Supplier#000005989 rjFY,5kgLpBu7c
Supplier#000006059 4m0cv8MwJ9yX2vlwI Z
Supplier#000006065 UiI2Cy3W4Tu5sLk
LuvXLRy6KihlGv
Supplier#000006070 TalC5m0pDrO6DZbngfmGmqe
Supplier#000006109 rY5gbfh3dKHnylcQUTPGCwnbe
Supplier#000006121
S92ycWwEzYYw4GspCBJN1WMuHhoZ
Supplier#000006215
j2iEbTsl,5PWdqWZ7k1yiISb7qtiiZljDIPEo
Supplier#000006217 RVN23SYT9jenUeaWGXUd
Supplier#000006274 S3yTZWqxTKUq g QQgcW9
AqhCkNZsW51hHuwU
Supplier#000006435 xIgE69XszYbnO4Eon7cHHO8y
Supplier#000006463 7 wkdj2EO49iotley2kmIM
ADpLSszGV3RNWj
Supplier#000006493 ojV
f,sNaB6Hm7r,fknDVTL63raJgAjZK
Supplier#000006521 b9 2zjHzxR
Supplier#000006607 3F 2e2gqD5u5B
Supplier#000006706
Ak4ga,ePu1QZ6C3qkrqjosaX0gxvqS9vkbe
Supplier#000006761
n4jhxGMqB5prD1HhpLvwrWStOLlla
Supplier#000006808 HGd2Xo 9nEcHJhZvXjXxWKIpApT
Supplier#000006858
fnlINT885vBBhsWwTGiZ0o22thwGY16h GHJj21
Supplier#000006872 XIDPiA7PLXCWK6SeEcld
Supplier#000006949 mLxYUJhsGcLtKe

```
,GFirNu183AvT
Supplier#000006985
PrUUiboQpy,OtgJ01Z4BxJQUyrw9c3I
Supplier#000007072          2tRyX9M1a
4Rcm57s779F1ANG9jlpK
Supplier#000007098
G3j8g0KC4OcbAu2OVoPHrXQWMCUdjq8wgCHOExu
Supplier#000007135          ls DoKV7V5ulfQy9V
Supplier#000007160
TqDGBULB3cTqIT6FKDvm9BS4e4v,zwYiQPb
Supplier#000007169          tEc95D2moN9S84nd55O,dlnW
Supplier#000007322          wr7dgte5q
MAjiY0uwmi3MyDkSMX1
Supplier#000007365          51xhROLvQMJ05DndtZWt
Supplier#000007398          V8eE6oZ00OFNU,
Supplier#000007402          4UVv58ery1rjmqSR5
Supplier#000007448          yhhpWiJi7EJ6Q5VCaQ
Supplier#000007477          9m9j0wfhWzCvVHxkU,PpAxwSH0h
Supplier#000007509          q8,V6LJRoHJjHcOuSG7aLTMg
Supplier#000007561          rMcFg2530VC
Supplier#000007789
rQ7cUcPrtudOyO3svNSkimqH6qrfWT2Sz
Supplier#000007801          69fi,U1r6enUb
Supplier#000007818          yhhc2CQec Jrvc8zqBi83
Supplier#000007885
u3sicchh5ZpyTUpN1cJKNcAoabIWgY
Supplier#000007918          r,v9mBQ6LoEYyj1
Supplier#000007926          ErzCF80K9Uy
Supplier#000007957          ELwnio14ssoU1 dRyZIL
OK3Vtzb
Supplier#000007965          F7Un5lJ7p5hhj
Supplier#000007968
DsF9UlZ2Fo6HXN9aErvyg1ikHoD582HSGZpP
Supplier#000007998
LnASFBfYRFOo9d6d,asBvVq9Lo2P
Supplier#000008168          aOa82a8ZbKCnfDLX
Supplier#000008231          IK7eGw Yj90sTdpsP,vcqWxLB
Supplier#000008243          2AyePMkDqmzVzjGTizXthFLo8h
EiudCMxOmIIG
Supplier#000008275          BlbNDfWg,gpXKQlLN
Supplier#000008323          75I18sZmASwm
POeheRMdj9tmpyeQ,BfCXN5BIAb
Supplier#000008366
h778cEj14BuW9OEKlvPTWq4iwASR6EBBXN7zeS8
Supplier#000008423
RQhKnkAhR0DAr3Ix4Q1weMMn00hNe Kq
Supplier#000008480          4sSDA4ACReklNjEm5T6b
Supplier#000008532
Uc29q4,5xVdDOF87UZrxhr4xWS0ihEUXuh
Supplier#000008595          MH0iB73GQ3z UW3O DbCbqmc
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 09/28/07 10:31:34.711041
Query Time =             2.2 secs


==================================================
```

qryqual21

```
==================================================
Start timestamp 09/28/07 10:33:47.643079

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

-- 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
Supplier#000003335                  15
Supplier#000005649                  15
Supplier#000006027                  15
Supplier#000006795                  15
Supplier#000006800                  15
Supplier#000006824                  15
Supplier#000007131                  15
Supplier#000007382                  15
Supplier#000008913                  15
Supplier#000009787                  15
Supplier#000000633                  14
Supplier#000001960                  14
Supplier#000002323                  14
Supplier#000002490                  14
Supplier#000002993                  14
Supplier#000003101                  14
Supplier#000004489                  14
Supplier#000005435                  14
Supplier#000005583                  14
Supplier#000005774                  14
Supplier#000007579                  14
Supplier#000008180                  14
Supplier#000008695                  14
Supplier#000009224                  14
Supplier#000000357                  13
Supplier#000000436                  13
Supplier#000000610                  13
Supplier#000000788                  13
Supplier#000000889                  13
Supplier#000001062                  13
Supplier#000001498                  13
Supplier#000002056                  13
Supplier#000002312                  13
Supplier#000002344                  13
Supplier#000002596                  13
Supplier#000002615                  13
Supplier#000002978                  13
Supplier#000003048                  13
Supplier#000003234                  13
Supplier#000003727                  13
Supplier#000003806                  13
Supplier#000004472                  13
Supplier#000005236                  13
Supplier#000005906                  13
Supplier#000006241                  13
Supplier#000006326                  13
Supplier#000006384                  13
Supplier#000006394                  13
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 09/28/07 10:34:55.250082
Query Time =            67.6 secs

================================================
```

qryqual22

```
================================================
Start timestamp 09/28/07 10:35:08.899802

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

-- 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 09/28/07 10:35:09.428342
Query Time =             0.5 secs
```

Appendix D. Seed and Query Substitution Parameters

This Appendix contains Seed values and substitution parameters for each stream

```
================================================
```

seed values

```
================================================

session 00  923011455
session 01  923011456
session 02  923011457
session 03  923011458
session 04  923011459
session 05  923011460
session 06  923011461
session 07  923011462
session 08  923011463

================================================
```

stream substitution parameters

power stream seed = 923011455

```
-- TPC TPC-H Parameter Substitution (Version 2.6.0
build 1)
-- using 923011455 as a seed to the RNG
Q1  DELTA        73
Q2  SIZE         43
    TYPE         TIN
    REGION       MIDDLE EAST
Q3  SEGMENT      HOUSEHOLD
    DATE         1995-03-31
Q4  DATE         1997-10-01
Q5  REGION       AFRICA
    DATE         1996-01-01
Q6  DATE         1996-01-01
    DISCOUNT     0.02
    QUANTITY     25
Q7  NATION1      ARGENTINA
    NATION2      UNITED KINGDOM
Q8  NATION       UNITED KINGDOM
    REGION       EUROPE
    TYPE         PROMO ANODIZED COPPER
Q9  COLOR        slate
Q10 DATE         1994-02-01
Q11 NATION       INDIA
    FRACTION     0.0000000333
Q12 SHIPMODE1    AIR
    SHIPMODE2    REG AIR
    DATE         1995-01-01
Q13 WORD1        special
    WORD2        accounts
Q14 DATE         1995-01-01
Q15 DATE         1993-01-01
Q16 BRAND        Brand#31
    TYPE         ECONOMY ANODIZED
    SIZE1        46
    SIZE2        42
    SIZE3        25
    SIZE4        5
    SIZE5        11
    SIZE6        4
    SIZE7        35
    SIZE8        15
Q17 BRAND        Brand#14
    CONTAINER    WRAP PACK
Q18 QUANTITY     314
Q19 BRAND1       Brand#54
    BRAND2       Brand#31
    BRAND3       Brand#44
    QUANTITY1    7
    QUANTITY2    16
    QUANTITY3    29
Q20 COLOUR       powder
    DATE         1995-01-01
    NATION       BRAZIL
Q21 NATION       JAPAN
Q22 I1           13
    I2           29
    I3           11
    I4           27
    I5           17
    I6           28
    I7           18
```

throughput stream = 1 seed = 923011456

```
-- TPC TPC-H Parameter Substitution (Version 2.6.0
build 1)
-- using 923011456 as a seed to the RNG
Q1  DELTA        81
Q2  SIZE         31
    TYPE         COPPER
    REGION       ASIA
Q3  SEGMENT      AUTOMOBILE
    DATE         1995-03-16
Q4  DATE         1995-07-01
Q5  REGION       AMERICA
    DATE         1996-01-01
Q6  DATE         1996-01-01
    DISCOUNT     0.07
    QUANTITY     25
Q7  NATION1      CHINA
    NATION2      MOROCCO
Q8  NATION       MOROCCO
    REGION       AFRICA
    TYPE         ECONOMY POLISHED COPPER
Q9  COLOR        saddle
Q10 DATE         1994-12-01
Q11 NATION       VIETNAM
    FRACTION     0.0000000333
Q12 SHIPMODE1    REG AIR
    SHIPMODE2    MAIL
    DATE         1996-01-01
Q13 WORD1        special
    WORD2        accounts
Q14 DATE         1996-01-01
Q15 DATE         1996-01-01
Q16 BRAND        Brand#11
    TYPE         STANDARD BURNISHED
    SIZE1        49
    SIZE2        39
    SIZE3        9
    SIZE4        38
    SIZE5        29
    SIZE6        6
    SIZE7        22
    SIZE8        28
Q17 BRAND        Brand#11
    CONTAINER    WRAP CAN
Q18 QUANTITY     312
Q19 BRAND1       Brand#51
    BRAND2       Brand#14
    BRAND3       Brand#44
    QUANTITY1    3
    QUANTITY2    17
    QUANTITY3    25
Q20 COLOUR       chartreuse
    DATE         1993-01-01
    NATION       MOZAMBIQUE
Q21 NATION       EGYPT
Q22 I1           17
    I2           16
    I3           25
    I4           31
    I5           24
    I6           22
    I7           29
```

throughput stream = 2 seed = 923011457

```
-- TPC TPC-H Parameter Substitution (Version 2.6.0
build 1)
-- using 923011457 as a seed to the RNG
Q1  DELTA        89
Q2  SIZE         18
    TYPE         BRASS
    REGION       AFRICA
Q3  SEGMENT      FURNITURE
    DATE         1995-03-02
Q4  DATE         1993-04-01
Q5  REGION       ASIA
    DATE         1996-01-01
```

```
Q6  DATE         1996-01-01
    DISCOUNT     0.05
    QUANTITY     24
Q7  NATION1      IRAN
    NATION2      GERMANY
Q8  NATION       GERMANY
    REGION       EUROPE
    TYPE         ECONOMY BURNISHED COPPER
Q9  COLOR        puff
Q10 DATE         1993-09-01
Q11 NATION       INDONESIA
    FRACTION     0.0000000333
Q12 SHIPMODE1    FOB
    SHIPMODE2    REG AIR
    DATE         1996-01-01
Q13 WORD1        special
    WORD2        accounts
Q14 DATE         1996-01-01
Q15 DATE         1994-01-01
Q16 BRAND        Brand#51
    TYPE         MEDIUM POLISHED
    SIZE1        3
    SIZE2        29
    SIZE3        44
    SIZE4        4
    SIZE5        12
    SIZE6        24
    SIZE7        13
    SIZE8        9
Q17 BRAND        Brand#13
    CONTAINER    SM BOX
Q18 QUANTITY     313
Q19 BRAND1       Brand#13
    BRAND2       Brand#52
    BRAND3       Brand#43
    QUANTITY1    8
    QUANTITY2    18
    QUANTITY3    21
Q20 COLOUR       midnight
    DATE         1997-01-01
    NATION       FRANCE
Q21 NATION       VIETNAM
Q22 I1           16
    I2           20
    I3           18
    I4           19
    I5           11
    I6           17
    I7           27
```

throughput stream = 3 seed = 923011458

```
-- TPC TPC-H Parameter Substitution (Version 2.6.0
build 1)
-- using 923011458 as a seed to the RNG
Q1  DELTA        97
Q2  SIZE         6
    TYPE         NICKEL
    REGION       ASIA
Q3  SEGMENT      MACHINERY
    DATE         1995-03-18
Q4  DATE         1995-11-01
Q5  REGION       MIDDLE EAST
    DATE         1997-01-01
Q6  DATE         1997-01-01
    DISCOUNT     0.02
    QUANTITY     25
Q7  NATION1      BRAZIL
    NATION2      UNITED STATES
Q8  NATION       UNITED STATES
    REGION       AMERICA
    TYPE         LARGE BRUSHED COPPER
Q9  COLOR        papaya
Q10 DATE         1994-06-01
Q11 NATION       RUSSIA
    FRACTION     0.0000000333
Q12 SHIPMODE1    MAIL
    SHIPMODE2    AIR
    DATE         1996-01-01
Q13 WORD1        special
    WORD2        accounts
Q14 DATE         1996-01-01
Q15 DATE         1996-01-01
Q16 BRAND        Brand#31
    TYPE         ECONOMY BRUSHED
    SIZE1        15
    SIZE2        38
    SIZE3        28
    SIZE4        3
    SIZE5        29
    SIZE6        44
    SIZE7        21
    SIZE8        24
Q17 BRAND        Brand#15
    CONTAINER    SM PACK
Q18 QUANTITY     315
Q19 BRAND1       Brand#15
    BRAND2       Brand#35
    BRAND3       Brand#32
    QUANTITY1    3
    QUANTITY2    19
    QUANTITY3    28
Q20 COLOUR       white
    DATE         1995-01-01
    NATION       VIETNAM
Q21 NATION       JORDAN
Q22 I1           21
    I2           28
    I3           17
    I4           15
    I5           32
    I6           29
    I7           14
```

throughput stream = 4 seed = 923011459

```
-- TPC TPC-H Parameter Substitution (Version 2.6.0
build 1)
-- using 923011459 as a seed to the RNG
Q1  DELTA        105
Q2  SIZE         44
    TYPE         TIN
    REGION       AFRICA
Q3  SEGMENT      FURNITURE
    DATE         1995-03-04
Q4  DATE         1993-08-01
Q5  REGION       AFRICA
    DATE         1997-01-01
Q6  DATE         1997-01-01
    DISCOUNT     0.08
    QUANTITY     25
Q7  NATION1      ROMANIA
    NATION2      MOZAMBIQUE
Q8  NATION       MOZAMBIQUE
    REGION       AFRICA
    TYPE         LARGE PLATED COPPER
Q9  COLOR        navajo
Q10 DATE         1993-03-01
Q11 NATION       IRAN
    FRACTION     0.0000000333
Q12 SHIPMODE1    TRUCK
    SHIPMODE2    AIR
    DATE         1996-01-01
Q13 WORD1        special
    WORD2        deposits
Q14 DATE         1996-01-01
Q15 DATE         1994-01-01
Q16 BRAND        Brand#21
    TYPE         SMALL BURNISHED
    SIZE1        9
    SIZE2        20
    SIZE3        15
    SIZE4        23
    SIZE5        26
    SIZE6        1
    SIZE7        40
    SIZE8        37
Q17 BRAND        Brand#12
    CONTAINER    SM CAN
Q18 QUANTITY     312
Q19 BRAND1       Brand#12
    BRAND2       Brand#23
    BRAND3       Brand#31
    QUANTITY1    8
    QUANTITY2    20
    QUANTITY3    25
Q20 COLOUR       hot
    DATE         1994-01-01
    NATION       IRAQ
Q21 NATION       ETHIOPIA
Q22 I1           21
    I2           15
```

```
    I3          14
    I4          19
    I5          34
    I6          13
    I7          30
```

throughput stream = 5 seed = 923011460

```
-- TPC TPC-H Parameter Substitution (Version 2.6.0
build 1)
-- using 923011460 as a seed to the RNG
Q1  DELTA       113
Q2  SIZE        32
    TYPE        COPPER
    REGION      EUROPE
Q3  SEGMENT     MACHINERY
    DATE        1995-03-20
Q4  DATE        1996-03-01
Q5  REGION      AMERICA
    DATE        1997-01-01
Q6  DATE        1997-01-01
    DISCOUNT    0.05
    QUANTITY    24
Q7  NATION1     IRAQ
    NATION2     INDIA
Q8  NATION      INDIA
    REGION      ASIA
    TYPE        LARGE BURNISHED TIN
Q9  COLOR       medium
Q10 DATE        1993-12-01
Q11 NATION      UNITED KINGDOM
    FRACTION    0.0000000333
Q12 SHIPMODE1   RAIL
    SHIPMODE2   AIR
    DATE        1997-01-01
Q13 WORD1       special
    WORD2       deposits
Q14 DATE        1997-01-01
Q15 DATE        1997-01-01
Q16 BRAND       Brand#51
    TYPE        LARGE PLATED
    SIZE1       38
    SIZE2       5
    SIZE3       47
    SIZE4       36
    SIZE5       19
    SIZE6       9
    SIZE7       20
    SIZE8       1
Q17 BRAND       Brand#13
    CONTAINER   LG BOX
Q18 QUANTITY    314
Q19 BRAND1      Brand#25
    BRAND2      Brand#51
    BRAND3      Brand#31
    QUANTITY1   4
    QUANTITY2   10
    QUANTITY3   21
Q20 COLOUR      salmon
    DATE        1997-01-01
    NATION      ALGERIA
Q21 NATION      UNITED KINGDOM
Q22 I1          12
    I2          20
    I3          21
    I4          27
    I5          24
    I6          29
    I7          28
```

throughput stream = 6 seed = 923011461

```
-- TPC TPC-H Parameter Substitution (Version 2.6.0
build 1)
-- using 923011461 as a seed to the RNG
Q1  DELTA       60
Q2  SIZE        19
    TYPE        STEEL
    REGION      AFRICA
Q3  SEGMENT     BUILDING
    DATE        1995-03-06
Q4  DATE        1993-12-01
Q5  REGION      ASIA
    DATE        1997-01-01
Q6  DATE        1997-01-01
    DISCOUNT    0.03
    QUANTITY    24
Q7  NATION1     CANADA
    NATION2     ALGERIA
Q8  NATION      ALGERIA
    REGION      AFRICA
    TYPE        MEDIUM BRUSHED TIN
Q9  COLOR       lemon
Q10 DATE        1994-10-01
Q11 NATION      IRAQ
    FRACTION    0.0000000333
Q12 SHIPMODE1   AIR
    SHIPMODE2   SHIP
    DATE        1994-01-01
Q13 WORD1       special
    WORD2       deposits
Q14 DATE        1997-01-01
Q15 DATE        1994-01-01
Q16 BRAND       Brand#31
    TYPE        PROMO BRUSHED
    SIZE1       27
    SIZE2       50
    SIZE3       31
    SIZE4       3
    SIZE5       19
    SIZE6       43
    SIZE7       45
    SIZE8       15
Q17 BRAND       Brand#15
    CONTAINER   LG PACK
Q18 QUANTITY    315
Q19 BRAND1      Brand#22
    BRAND2      Brand#44
    BRAND3      Brand#25
    QUANTITY1   9
    QUANTITY2   11
    QUANTITY3   28
Q20 COLOUR      dark
    DATE        1996-01-01
    NATION      MOROCCO
Q21 NATION      MOROCCO
Q22 I1          33
    I2          19
    I3          31
    I4          26
    I5          15
    I6          22
    I7          12
```

throughput stream = 7 seed = 923011462

```
-- TPC TPC-H Parameter Substitution (Version 2.6.0
build 1)
-- using 923011462 as a seed to the RNG
Q1  DELTA       68
Q2  SIZE        7
    TYPE        NICKEL
    REGION      EUROPE
Q3  SEGMENT     MACHINERY
    DATE        1995-03-22
Q4  DATE        1996-07-01
Q5  REGION      EUROPE
    DATE        1997-01-01
Q6  DATE        1997-01-01
    DISCOUNT    0.08
    QUANTITY    25
Q7  NATION1     SAUDI ARABIA
    NATION2     PERU
Q8  NATION      PERU
    REGION      AMERICA
    TYPE        MEDIUM PLATED TIN
Q9  COLOR       indian
Q10 DATE        1993-07-01
Q11 NATION      UNITED STATES
    FRACTION    0.0000000333
Q12 SHIPMODE1   SHIP
    SHIPMODE2   AIR
    DATE        1997-01-01
Q13 WORD1       pending
    WORD2       deposits
Q14 DATE        1997-01-01
Q15 DATE        1997-01-01
Q16 BRAND       Brand#21
    TYPE        MEDIUM ANODIZED
    SIZE1       19
    SIZE2       33
    SIZE3       18
```

```
    SIZE4       11                                          I5          22
    SIZE5       27                                          I6          16
    SIZE6       8                                           I7          25
    SIZE7       21
    SIZE8       14
Q17 BRAND       Brand#12
    CONTAINER   LG DRUM
Q18 QUANTITY    313
Q19 BRAND1      Brand#24
    BRAND2      Brand#22
    BRAND3      Brand#24
    QUANTITY1   4
    QUANTITY2   12
    QUANTITY3   24
Q20 COLOUR      orchid
    DATE        1994-01-01
    NATION      ETHIOPIA
Q21 NATION      GERMANY
Q22 I1          15
    I2          30
    I3          14
    I4          19
    I5          29
    I6          32
    I7          23
```

throughput stream = 8 seed = 923011463

```
-- TPC TPC-H Parameter Substitution (Version 2.6.0
build 1)
-- using 923011463 as a seed to the RNG
Q1  DELTA       76
Q2  SIZE        45
    TYPE        TIN
    REGION      AMERICA
Q3  SEGMENT     BUILDING
    DATE        1995-03-08
Q4  DATE        1994-04-01
Q5  REGION      MIDDLE EAST
    DATE        1993-01-01
Q6  DATE        1993-01-01
    DISCOUNT    0.05
    QUANTITY    24
Q7  NATION1     JAPAN
    NATION2     INDONESIA
Q8  NATION      INDONESIA
    REGION      ASIA
    TYPE        MEDIUM ANODIZED TIN
Q9  COLOR       ghost
Q10 DATE        1994-04-01
Q11 NATION      JAPAN
    FRACTION    0.0000000333
Q12 SHIPMODE1   FOB
    SHIPMODE2   RAIL
    DATE        1997-01-01
Q13 WORD1       pending
    WORD2       deposits
Q14 DATE        1997-01-01
Q15 DATE        1995-01-01
Q16 BRAND       Brand#51
    TYPE        ECONOMY PLATED
    SIZE1       23
    SIZE2       4
    SIZE3       10
    SIZE4       2
    SIZE5       18
    SIZE6       20
    SIZE7       45
    SIZE8       13
Q17 BRAND       Brand#14
    CONTAINER   MED BOX
Q18 QUANTITY    314
Q19 BRAND1      Brand#31
    BRAND2      Brand#54
    BRAND3      Brand#23
    QUANTITY1   9
    QUANTITY2   13
    QUANTITY3   20
Q20 COLOUR      black
    DATE        1997-01-01
    NATION      ROMANIA
Q21 NATION      UNITED STATES
Q22 I1          29
    I2          11
    I3          17
    I4          14
```

Appendix E. Implementation-Specific Layer/Driver Code

==========

auditTest.sh

```
#!/bin/ksh
setupDir
db2start
buildtpcd > buildtpcd.out
runpower UF > runpower.1.out
runthroughput UF > runthroughput.1.out
setupRun
runpower UF > runpower.2.out
runthroughput UF > runthroughput.2.out
```

==========

runpower

```
:   # -*-Perl-*-
##########################################################
######################
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use,
duplication or
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
##########################################################
######################
eval 'exec perl5 -S $0 ${1+"$@"}' # Horrible kludge to
convert this
    if 0;                         # into a "portable"
perl script

# usage  runpower [UF]
# where UF is the optional parameter that says to run
the power test
# with the update functions.  By default, the update
functions are not
# run

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

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

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

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

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

if (length($ENV{"TPCD_AUDIT_DIR"}) <= 0)
{
  die "TPCD_AUDIT_DIR environment variable not set\n";
}
if (length($ENV{"TPCD_RUN_DIR"}) <= 0)
{
  die "TPCD_RUN_DIR environment variable not set\n";
}
if (length($ENV{"TPCD_DBNAME"}) <= 0)
{
  die "TPCD_DBNAME environment variable not set\n";
}
if (length($ENV{"TPCD_RUNNUMBER"}) <= 0)
{
  die "TPCD_RUNNUMBER environment variable not set\n";
}
if (length($ENV{"TPCD_SF"}) <= 0)
{
  die "TPCD_SF environment variable not set\n";
}
if (length($ENV{"TPCD_PLATFORM"}) <= 0)
{
  die "TPCD_PLATFORM environment variable not set\n";
}
if (length($ENV{"TPCD_PATH_DELIM"}) <= 0)
{
  die "TPCD_PATH_DELIM environment variable not
set\n";
}
if (length($ENV{"TPCD_PRODUCT"}) <= 0)
{
  die "TPCD_PRODUCT environment variable not set\n";
}
if (length($ENV{"TPCD_AUDIT"}) <= 0)
{
    die "Must set TPCD_AUDIT env't var.  Real audit
timing sequence run if yes\n";
}
if (length($ENV{"TPCD_PHYS_NODE"}) <= 0)
{
    die "TPCD_PHYS_NODE env't var not set\n";
}
if (length($ENV{"TPCD_LOG_DIR"}) <= 0)
{
    $ENV{"TPCD_LOG_DIR"} = "NULL";
}
if (length($ENV{"TPCD_MODE"}) <= 0)
{
  die "TPCD_MODE environment variable not set -
uni/smp/mln \n";
}
if (length($ENV{"TPCD_ROOTPRIV"}) <= 0)
{
  die "TPCD_ROOTPRIV environment variable not set -
yes/no \n";
}

#set up local variables
$runNum=$ENV{"TPCD_RUNNUMBER"};
$runDir=$ENV{"TPCD_RUN_DIR"};
$auditDir=$ENV{"TPCD_AUDIT_DIR"};
$dbname=$ENV{"TPCD_DBNAME"};
$sf=$ENV{"TPCD_SF"};
$platform=$ENV{"TPCD_PLATFORM"};
$delim=$ENV{"TPCD_PATH_DELIM"};
$gatherstats=$ENV{"TPCD_GATHER_STATS"};
$product=$ENV{"TPCD_PRODUCT"};
$RealAudit=$ENV{"TPCD_AUDIT"};
$inlistmax=$ENV{"TPCD_INLISTMAX"};
$pn=$ENV{"TPCD_PHYS_NODE"};
$logDir=$ENV{"TPCD_LOG_DIR"};
$rootPriv=$ENV{"TPCD_ROOTPRIV"};
$mode=$ENV{"TPCD_MODE"};
if (( $mode eq "uni" ) || ( $mode eq "smp" ))
{
  $all_ln="once";
  $all_pn="once";
  $once="once";
}
else
{
  $all_ln="all_ln";
  $all_pn="all_pn";
  $once="once";
}


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

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

# append isolation level information about tpcdbatch
to the miso file
# the miso file is created here but appended to for
power and throughput
#information

$misofile="$runDir${delim}miso$runNum";
if ( -e $misofile )
{
    &rm("$misofile");
}
# if we are in real audit mode then we must start the
db manager now since
# there must be no activity on the database between
the time the build script
# has finished and the time the power test is started
if ( $RealAudit eq "yes" )
{
    system("db2start");
    system("db2 activate database $dbname");
}

# do not activate the database
#if ( $RealAudit ne "yes" )
#{
#    system("db2 activate database $dbname");
#}



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

open(MISO, ">$misofile") || die "Can't open $misofile:
$!\n";
$curTs = `perl gettimestamp "long"`;
print MISO "Timestamp and isolation level of tpcdbatch
before power run at : $curTs\n";
close(MISO);
if ( $product eq "pe" )
{
    system("db2 \"connect to $dbname\"; db2 \"select
name,creator,valid,unique_id,isolation from
sysibm.sysplan where name like 'TPCD%'\"; db2 connect
reset; db2 terminate  >> $runDir${delim}miso$runNum
");
}
else
{
    &verifyTPCDbatch("$misofile","$dbname");
}

if ($platform eq "aix")
{

  # Create the sysunused file.  This reports what
disks are attached, and which
  # ones are being used.  Its use spans both the
runpower and 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";
print "$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\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 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}mpstrcn
t$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${delim}mpinte
r$runNum.metrics");

if ($runUF eq "UF") {
  &cat("$runDir${delim}mpufinter*","$runDir${delim}mpi
nter$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${runN
um}";
  open(DBTUNE, ">$dbtunefile") || die "Can't open
$dbtunefile: $!\n";
  $timestamp=`perl gettimestamp "long"`;
  print DBTUNE "Database and Database manager
configuration taken at : $timestamp";
  close(DBTUNE);
  system("db2level >> $dbtunefile");
  system("db2 get database configuration for $dbname
>> $dbtunefile");
  system("db2 get database manager configuration >>
$dbtunefile");
  system("db2set >> $dbtunefile");
```

```
  if (( $mode eq "mln" ) || ( $mode eq "mpp"))
  {
     $cfgfile="$runDir${delim}dbtune${runNum}.";
     #removed by Alex due to hang
     #system("db2_all '||\" typeset -i ln=##; db2 get
db cfg for $dbname > $cfgfile\${ln} ; db2 get dbm cfg
>> $cfgfile\${ln}; db2set >> $cfgfile\${ln}; db2
terminate '");
  }

}

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

}

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

====================================================
```

runthroughput

```
:   # -*-Perl-*-
##########################################################
######################
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use,
duplication or
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
##########################################################
######################
#eval 'exec perl5 -S $0 ${1+"$@"}' # Horrible kludge
to convert this
eval 'exec perl -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}tpc
dbatch -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}tpc
dbatch -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"`;
        }
        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${delim}mtint
er$runNum.metrics");

if ($runUF ne "no") {
  &cat("$runDir${delim}mtufinter*","$runDir${delim}mti
nter$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${runN
um}";
  open(DBTUNE, ">$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");
}
#
```

ploaduf1

```
#!/bin/ksh
#########################################################
######################
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use,
duplication or
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
#########################################################
########################

RFpair=$1
~/tpch/tools/load_line_uf $RFpair &
~/tpch/tools/load_orders_uf $Rfpair
```

ploaduf2

```
#!/bin/ksh
#########################################################
######################
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use,
duplication or
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
#########################################################
########################

RFpair=$1;
db2 connect to tpcd
db2 "load from delete.${RFpair}.new of del modified by
coldel| fastparse  messages /dev/null replace into
TPCDTEMP.ORDERS_DEL nonrecoverable partitioned db
config mode load_only part_file_location
/tpch/updates;"
db2 commit;
db2 connect reset
db2 terminate;
```

load_line_uf

```
#!/bin/ksh
#########################################################
######################
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use,
duplication or
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
#########################################################
########################

RFpair=$1;
db2 connect to tpcd
db2 "load from lineitem.tbl.u${RFpair}.new of del
modified by coldel| fastparse  messages /dev/null
replace into TPCDTEMP.LINEITEM_new nonrecoverable
partitioned db config mode load_only
part_file_location /tpch/updates;"
db2 commit;
db2 connect reset
db2 terminate
```

load_orders_uf

```
#!/bin/ksh
#########################################################
######################
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use,
duplication or
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
#########################################################
########################

RFpair=$1;
db2 connect to tpcd
db2 "load from orders.tbl.u${RFpair}.new of del
modified by coldel| fastparse  messages /dev/null
replace into TPCDTEMP.ORDERS_new nonrecoverable
partitioned db config mode load_only
part_file_location /tpch/updates; "
db2 commit;
db2 connect reset
db2 terminate
```

calcmetrics.pl

```
:   # -*-Perl-*-
#########################################################
######################
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
```

```
## US Government Users Restricted Rights - Use,
duplication or
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
#########################################################
########################
eval 'exec perl5 -S $0 ${1+"$@"}' # Horrible kludge to
convert this
    if 0;                         # into a "portable"
perl script

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

require 'timelocal.pl';
require 'ctime.pl';
require 'getvars';
require 'macro.pl';
$numQueries=$ENV{"TPCD_NUM_QUERIES"};
$platform=$ENV{"TPCD_PLATFORM"};
$workload=$ENV{"TPCD_WORKLOAD"};
$NumQ = 0;
if (@ARGV > 0)
{
    $runUF=$ARGV[0];
}
else
{
    $runUF="no";
}

###################################
# gen_test
#
# AUTHOR: Jennifer Crawford
#         jenn @ torolab2
#
# IBM INTERNAL USE ONLY
###################################

#########################################################
######################
# This Perl script generates the Qth and Qph values
for the current
# throughput run.  It bases the calculations on the
data in
# mstrcnt?  It also uses the Qpp
# information from the mpinter file.
# Other information it needs are the number of streams
(TPCD_NUMSTREAM)
# and the scale factor (TPCD_SF).
# When invoking, optionally pass the directory to
search for these files in
# as parameter 1.  If this parameter is not defined
then the environment
# variables are used.
#########################################################
######################

$minStartYear = 99;
$minStartMonth = 99;
$minStartDay = 99;
$minStartHour = 99;
$minStartMinute = 99;
$minStartSecond = 99;
$maxStopYear = 00;
$maxStopMonth = 00;
$maxStopDay = 00;
$maxStopHour = 00;
$maxStopMinute = 00;
$maxStopSecond = 00;

$minStartTime=999999999999;
$maxStopTime=000000000000;

$n='\d\d';
if ($platform eq "nt")
{
    $m='\d\d\d';
}
elsif ( $platform eq "aix" || $platform eq "sun" ||
$platform eq "ptx" || $platform eq "hp" || $platform
eq "linux")
{
    $m='\d\d\d\d\d\d';
}
else
{
    die "Platform $platform not supported in
calcmetrics.pl\n";
}

$timestamp = "($n)/($n)/($n) ($n):($n):($n)\.($m)";

#paramter is the number of streams that the run
has....if this is
#not supplied then the default is the environment
variable TPCD_NUMSTREAM
if ( @ARGV > 0 )
{
    $numStreams = $ARGV[0];
}
else
{
    $numStreams = 0;
}
# comment this argument checking out...doesn't work
well when
# called from runthroughput now that runthroughpout
takes a parameter
#if ((defined $ARGV[0]))
#{
#       if (-d $ARGV[0])
#       {
#               $filepath=$ARGV[0];
#       }
#       else
#       {
#               print STDERR "$ARGV[0] is not a valid
directory\n\n";
#               print STDERR "usage: calcmetrics.pl
[directory]\n";
#               print STDERR "where directory is the
path to the mpinter files\n";
#               print STDERR "and the mstrcnt?
files,\n";
#               exit(1);
#       }
#}
#else
#{
        $audit_dir = $ENV{"TPCD_AUDIT_DIR"};
        $run_num = $ENV{"TPCD_RUNNUMBER"};
        $filepath = $ENV{"TPCD_RUN_DIR"};
  if ( $numStreams == 0 )
  {
    $numStreams = $ENV{"TPCD_NUMSTREAM"};
  }
  $SF = $ENV{"TPCD_SF"};
  $dbname = $ENV{"TPCD_DBNAME"};
#}

open(MSTRCNT_FILELIST,"$dircmds
$filepath${delim}mstrcnt* |"); #time stamp file
open(MP_FILELIST,"$dircmds $filepath${delim}mpinter*
|");  # runpower metrics file
open(MTQ_FILELIST,"$dircmds
$filepath${delim}mts1inter* |"); #only need the first
one

print "Using Directory = $filepath\n";


print "Processing throughput stream count files:\n";

while(<MSTRCNT_FILELIST>)
{
        $filename = $_;
        chop($filename);
        if ( $platform eq "nt" )
        {
          $filename = $filepath."\\".$filename;
        }
        print "$filename\n";

  open(IN,$filename);
  while(<IN>)
  {
    chop;

    if (m@^.* .* .* starting at $timestamp$@)
    {
      $startYear = $3;
```

```
      $startMonth = $1;
      $startDay = $2;
      $startHour = $4;
      $startMinute = $5;
      $startSecond = $6;
      $startFrac = $7;
      $startTime =
&timegm($startSecond,$startMinute,$startHour,$startDay
,($startMonth - 1),$startYear);

      $startTime = "$startTime\.$startFrac";
      # if this current start time is sooner than the
previous min start time
      # update the minimum
      if ( $startTime < $minStartTime )
      {
          $minStartTime = $startTime;
          $minStartYear = $startYear;
          $minStartMonth = $startMonth;
          $minStartDay = $startDay;
          $minStartHour = $startHour;
          $minStartMinute = $startMinute;
          $minStartSecond = $startSecond;
       }
    }
    if (m@^.* .* .* stopping at $timestamp$@)
    {
      $stopYear = $3;
      $stopMonth = $1;
      $stopDay = $2;
      $stopHour = $4;
      $stopMinute = $5;
      $stopSecond = $6;
      $stopFrac = $7;
      $stopTime =
&timegm($stopSecond,$stopMinute,$stopHour,$stopDay,($s
topMonth - 1),$stopYear);

      $stopTime = "$stopTime\.$stopFrac";
      # if this current stop time is later than the
previous max stop time
      # update the maximum
      if ( $stopTime > $maxStopTime )
      {
          $maxStopTime = $stopTime;
          $maxStopYear = $stopYear;
          $maxStopMonth = $stopMonth;
          $maxStopDay = $stopDay;
          $maxStopHour = $stopHour;
          $maxStopMinute = $stopMinute;
          $maxStopSecond = $stopSecond;
       }
    }
  }
  close(IN);
}

#now calculate the elapsed time in seconds
$Ts = $maxStopTime - $minStartTime;
$Ts = sprintf("%.0f", $Ts);

print "Processing throughput query file\n";
while(<MTQ_FILELIST>)
{
       $filename = $_;
       chop($filename);
       if ( $platform eq "nt" )
       {
         $filename = $filepath."\\".$filename;
       }
       print "$filename\n";

  open(IN,$filename);
  while(<IN>)
  {
    chop;
    if (/^Number of statements: (.*)$/)
    {
           $NumQ = $1;
    }

  }
  close(IN);
}

$Qthtop = $numStreams * $NumQ * 3600 * $SF;
$Qth = $Qthtop / $Ts;

print "Processing power file\n";
while(<MP_FILELIST>)
{
       $filename = $_;
       chop($filename);
       if ( $platform eq "nt" )
       {
         $filename = $filepath."\\".$filename;
       }
       print "$filename\n";

  open(IN,$filename);
  while(<IN>)
  {
    chop;

    if (/^Qpp${workload}.* = (.*)$/)
    {
        $Qpp = $1;
    }
  }
  close(IN);
}

$Qph = sqrt($Qpp * $Qth);

# Determine the size of the database from the scale
factor (1 SF = 1GB)
   if ($SF < 1.0) {
        $db_size = $SF * 1000;
        $db_size_qualifier = "MB";
   } elsif ($SF >= 1000.0) {
        $db_size = $SF / 1000;
        $db_size_qualifier = "TB";
   } else {
        $db_size = $SF;
        $db_size_qualifier = "GB";
   }


# open the output file and print the results
$outfilename =
"$filepath${delim}mtinter$run_num.metrics";
open(OUT,"> $outfilename");

print OUT &ctime(time), "\n";
print OUT "The TPC-${workload} metrics for database
$dbname at scale factor $SF are:\n";
print OUT "Ts         = $Ts\n";
printf (OUT "Qpp%s@%.1f%s = %f\n",$workload, $db_size,
$db_size_qualifier,$Qpp);
printf (OUT "Qth%s@%.1f%s = %f\n",$workload, $db_size,
$db_size_qualifier,$Qth);
printf (OUT "Qph%s@%.1f%s = %f\n",$workload, $db_size,
$db_size_qualifier,$Qph);

close(OUT);

# Write update stats to update output files
if($runUF ne "no")
{
   system("perl mupdate.pl t");
}
```

getvars

```
#!/usr/bin/perl
##########################################################
######################
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use,
duplication or
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
```

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

#################################
# 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");

&setVar(@reqVars, "ERROR");

$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.

open(SETTINGS, "tpcd.runsetup");
&parseVars(SETTINGS);
close (SETTINGS);

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";
        }
}
1
```

macro.pl

```
#!/usr/bin/perl
#########################################################
#######################
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use,
duplication or
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
#########################################################
#######################

# 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(($_='').'select($_),$|
=0),@_)'-ee,@_)[$[+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(($_='').'select($_),$|
=1),@_)'-ee,@_)[$[+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","rtestd
b","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 ALWAYS returns 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 SQLTAG.NAM
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,$ju
nk) = 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\\incl
ude\\rsa;.;t:\\msvc\\include;%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\\incl
ude\\rsa;.;t:\\msvc\\include;%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 \@l.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\\incl
ude\\rsa;.;t:\\msvc\\include;%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\\incl
ude\\rsa;.;t:\\msvc\\include;%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
#    <+|->#  - 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
##########################################################
########################
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use,
duplication or
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
##########################################################
########################
# 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") {
# running with ";" background causing hangups on some
servers
#       $ret=system("db2_all \'; db2 connect to
${dbname}; ${cmd}; db2 connect reset; db2 terminate \'
");
       $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" ) {
# running with ";" background causing hangups on some
servers
#       $ret=system("db2_all \'; ${cmd} \' ");
       $ret=system("db2_all \'${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" ) {
# running with ";" background causing hangups on some
servers
#        $ret=system("db2_all \'; db2 connect to
${dbname}; db2 $db2options -f $fname; db2 connect
reset; db2 terminate \' ");
        $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" ) {
# running with ";" background causing hangups on some
servers
#        $ret=system("db2_all \'; db2 $db2options -f
$fname; db2 terminate \' ");
        $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;
```

buildmlog

```
#!/usr/bin/perl
############################################################
######################
## Licensed Materials - Property of IBM
##
## Governed under the terms of the International
## License Agreement for Non-Warranted Sample Code.
##
## (C) COPYRIGHT International Business Machines Corp.
1990 - 2005
## All Rights Reserved.
##
## US Government Users Restricted Rights - Use,
duplication or
## disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
############################################################
######################
# usage  buildmlog
```

```
require 'getvars';

#set up local variables
$runNum=$ENV{"TPCD_RUNNUMBER"};
$auditDir=$ENV{"TPCD_AUDIT_DIR"};
$runDir=$ENV{"TPCD_RUN_DIR"};
$dbname=$ENV{"TPCD_DBNAME"};
$sf=$ENV{"TPCD_SF"};
$product=$ENV{"TPCD_PRODUCT"};
$home=$ENV{"HOME"};

$dblogstmt=`db2 "get database configuration for
$dbname" | grep "Path to log files" | cut -d"=" -f2`;

$curTs = `perl gettimestamp "long"`;
system("echo \"$curTs\" > $runDir/mlog$runNum");

if ( $product eq "pe" )
{
  #for pe we must substitute the NODE0000x with NODE*
so we can query all nodes
  $dblogpath=`db2 "get database configuration for
$dbname" | grep "Path to log files" | cut -d"=" -f2 |
sed 's/NODE0..../NODE\*/g'`;
  print "I don't know if the syntax for the next line
works on pe!!!\n";
  system("db2_all \"ls -lt $dblogpath\" >>
$runDir/mlog$runNum");
}
else
{
  $dblogpath=`db2 "get database configuration for
$dbname" | grep "Path to log files" | cut -d"=" -f2`;
  system("(ls -lt $dblogpath) >>
$runDir/mlog$runNum");
}

system("echo \"The previous list showed the log
directory size for the database\" >>
$runDir/mlog$runNum");
system("echo \"To compute the amount of log for 8
hours of run time, please use\" >>
$runDir/mlog$runNum");
system("echo \"the following formula:   \" >>
$runDir/mlog$runNum");
system("echo \"  logsize = 8.5MB * TPCD_SF *
TPCD_RUNNUMBER\"  >> $runDir/mlog$runNum");
$logsize=8.5 * $sf * $runNum;
system("echo \"  logsize = $logsize MB\"  >>
$runDir/mlog$runNum");

if ( $product eq "pe" )
{
  # for pe we must take the amount of space per node
to be the total log
  # for this we must determine the number of nodes in
the configuration.
  # the nodes are defined in the db2nodes.cfg file
  $numNodes=`wc -l $home/sqllib/db2nodes.cfg`;
  # this next strange thing changes the string that we
got from wordcount
  # (something like "   4   path/sqllib/db2nodes.cfg
").  It matches
  # the initial blanks, the first word (4) (which it
keeps) and the remainder
  # and substitutes the whole thing for the first word
(4)...maybe obvious
  $numNodes=~ s/\s*(\w*)\W.*/\1/;
  $perNodeLogSize= $logsize / $numNodes;
  system("echo \"  Number of nodes = $numNodes\" >>
$runDir/mlog$runNum");
  system("echo \"  logsize per node = $perNodeLogSize
MB per node\" >> $runDir/mlog$runNum");
}

1;
```

tpcdbatch.sqc

```
/*
########################################################
####################### */
/* ## Licensed Materials - Property of IBM
*/
/* ##
*/
/* ## Governed under the terms of the International
*/
/* ## License Agreement for Non-Warranted Sample Code.
*/
/* ##
*/
/* ## (C) COPYRIGHT International Business Machines
Corp. 1990 - 2005            */
/* ## All Rights Reserved.
*/
/* ##
*/
/* ## US Government Users Restricted Rights - Use,
duplication or                 */
/* ## disclosure restricted by GSA ADP Schedule
Contract with IBM Corp.          */
/*
########################################################
####################### */
/*******************************************************
**********************
*
*   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 count
*                (maxwait) was incremented on every
execution of the stmt as
*                opposed to just when deadlock really
happened.
*  14 nov 97 jen jenSEM - fix up error reporting on
semaphore failure
*                sem_op now returns failure to caller
so caller can report where
*                failure has happened.
*                Forced dbname to be upper case, an
all other parts of update
*                pair number to be lowercase
*  15 nov 97 jen SEED Reworked code to grab the seed
from the seed file.  Now
*                reusing seeds between runs, so power
run will always use first
*                seed, throughput will use the 2nd -
#stream+1 seeds
*
*  13 jan 98 jen LONG  Increase stmt_str to be able to
hold inlists with larger
*                order key numbers
*  04 mar 98 jen IMPORT added support for
TPCD_UPDATE_IMPORT to chose whether
*                using import or load api's for
loading data into the staging
*                tables
*  04 mar 98 jen TIMER changed from using gettimer to
gettimeofday for unix
*  01 apr 98 jen Fixed IMPORT code to do the proper
checking on strcmp (ie !strcmp)
*  01 apr 98 jen removed code to handle vldb - not
needed
*                Upgraded version to 2.4         for (
chunk
*  01 apr 98 jen Fixed up import code on NT so the
variable is recognized in the
*                children
*  25 may 98 sks Reworked some of the environment
variable code so consolidate as
*                much as possible.  Not all complete
because of differences in
*                the way nt and AIX calls (and starts
stuff in background) for UFs
*  29 may 98 jen REUSE_STAGE Changed UF1 so we reuse
the same staging tables
*                instead of having a new set for each
update pair
*  06 jul 98 jen Removed locking of staging tables
since they are created with
*                locksize table now
*  06 jul 98 jen 912RETRY - added code to retry query
execution on 912 as well
*                as 911
*  07 jul 98 jen Fixed summary_table() so 1000x
adjustment not based on UF (setting
*                of max and min pointers
*                Added generic SleepSome function to
handle NT vs AIX sleep differences
*  01 apr 98 djd Added change to permit the use of
table functions for UF1.
*                to enable this set TPCD_UPDATE_IMPORT
to tf in TPCD.SETUP file.
*                MERGED this into base copy on Jul 07
*  10 jul 98 jen haider's fix for 'outstream' var for
error processing in
*                runUF1_fn and runUF2_fn
*                Updated version to 2.5
*  25 sep 98 jen Added stream number printing into
mpqry* files and increases
*                accuracy of timestamp in mpqry (and
mts*qry*) files
*  06 oct 98 jen TIME_ACC Added accuracy of timestamp
in mpqry (and mts*qry*)
*                files. Cleaned up misuse of Sleep and
flushed buffers on
*                deadlocks
*  19 oct 98 kbs fix UF2_fn to correctly count rows
deleted in case of deadlock
*  20 oct 98 kbs rewrite UF2 and UF2_fn for static SQL
with staging table
*  23 oct 98 jen Cleaned up retrying of order/lineitem
on lineitem deadlock in UF1
*  24 oct 98 jen Used load_uf1 and load_uf2 instead of
general load_updates
*  26 oct 98 kbs inject the UF1 with a single staging
table
*  02 nov 98 jen Fixed processing of multiple chunks
in uf2 so don't duplicate
*  21 nov 98 kmw Fixed BIGINT
*  05 dec 98 aph Moved runUF1_fn() and runUF2_fn()
into a separate file tpcdUF.sqc
*                so that it can be bound separately
with a different isolation level.
*  21 dec 98 aph Integrated Jennifer's QppD
calculation (rounding & adjustment) fixes.
*  22 dec 98 aph For UFs during Throughput run, defer
CONNECT until children launched.
*  28 dec 98 aph Removed error_check() call after
CONNECT RESET
*  29 dec 98 aph For UFs do not COMMIT in
tpcdbatch.sqc.  COMMITs happen in tpcdUF.sqc.
*  18 jan 99 kal replaced header with #include
"tpcdbatch.h"
*  27 may 99 bbeaton from (03 mar 99 jen) Fixed SUN
fix that wasn't compatible with
*                NT (using %D %T instead of %x %X for
strftime)
*  16 jun 99 jen Added missing LPCTSTR cast of
semaphore file name for NT
*  17 jun 99 jen SEMA Changes semaphore file for
update functions to look for tpcd.setup
*                not for the orders.***  update data
file
*  21 jul 99 bbeaton Added semaphore control that
allows runpower to be run as two
*                separate streams (update and query).
This involves the use of
*                two semaphores to be used as it
executes in three different
*                sections.  The first is the update
inserts.  The next is the query
*                stream which is started with the
update stream, but waits until
*                the inserts are complete.  The third
section is the update deletes
*                which execute after the queries are
complete.
*  21 jul 99 bbeaton Added functions to handle
semaphore creation, control, etc.
*  21 jul 99 bbeaton Modified output to mp*inter
files.  It now only outputs
*                intermediate data that will be
calculated by 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 */
   int                  qnum;                  /*
current query number        */
   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 */
   char                 *vmstat_out_path;      /*
pathname of vmstat output file */
   char                 *iostat_out_path;      /*
pathname of iostat output file */
   int                  run_encountered_error; /*
whether run encountered error: -ve SQLCODE or 0 rows
fetched */
};


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

struct stmt_info
{
   long                 max_rows_fetch;
   long                 max_rows_out;
#ifdef TPCD_PROGRESS_FILE
   long                 sqlcode;               /*
sqlcode                  */
   long                 rows_fetch;            /*
number of rows returned    */
#endif
   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 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             */
#ifdef TPCD_PROGRESS_FILE
FILE            *progress_file;
#endif
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 */
#ifdef SQLWINT
int             pupdatePair;
BOOL            Parallel_Load_Complete;
int             iParallel_Load_Result;
LPVOID          ThreadParm;
DWORD           Thread_Parallel_Load_ID;
#endif
double          inter_uf_child_delay;   /*  delay
between starting successive UF's in seconds */

#ifdef SQLWINT
#define NO_PARALLEL_FORMAT_FETCH  /*   no parallel
format-fetch on windows */
#endif
#ifndef NO_PARALLEL_FORMAT_FETCH  /*   parallel
format-fetch only if requested */
/*************************************************
************************************
 *      fields relating to parallel processing, using
a slave_formatter process
 *      to format the data retrieved by the main
parent process in parallel with
 *      the parent running the queries and storing the
raw output in shared memory.
 *      structure of shared_mem area:
 *          .   first buffer
 *          .   second buffer
 *          .   control fields
```

```
 *      The processes flip-flop between buffers, one
storing into one while the other
 *      formats the other.   The control fields are
used to communicate between processes.
 *      Each buffer consists of three parts:
 *           header             lines of printable
text preceding the rows, e.g. :
 *                                Start timestamp
mm/dd/yy hh:mm:ss.uuuuuu
 *                                query tag and
statement text
 *           row data           binary row data header
consisting of sqld and column formatting lengths,
 *                                and then, for
each row, for each column in row, type||leng||data
 *         trailer              lines of printable
text followining the rows, e.g. :
 *                                Number of rows
retrieved
 *                                Stop timestamp
mm/dd/yy hh:mm:ss.uuuuuu
 *      Some synchronization between processes is
needed:
 *      . slave must wait until all data from a query
is fetched before starting to format
 *      . parent must wait before starting new query
if slave is still formatting the
 *        previous one.
 *      These are designated as rendezvous.  When a
process reaches a rendezvous,
 *      it must obey the following rule:
 *       if the other process is waiting, then wake it
up and proceed, else wait.
 */
#ifndef SLAVE_SEM
 /*      signals are used to tell a waiting process to
proceed.                          */
#else /* SLAVE_SEM */
 /*      semaphores are used to tell a waiting process
to proceed.                       */
#endif /* SLAVE_SEM */
/*      A single field in shared-mem indicates whether
either no process is waiting,
 *      or parent is waiting for slave, or slave is
waiting for parent.
 *      (parent should ideally never have to wait for
slave).
 *      This control field is updated using atomic
compare_and_swap so that the two
 *      processes cannot both put themselves into a
wait and hang.
 ************************************************
************************************/

#ifndef SLAVE_SEM
sigset_t emptymask, blockusermask, prevmask, tempmask;
typedef struct sigaction SIGACTYPE;
SIGACTYPE chsigaction, olsigaction;
int             signalled;          /*  whether
process has been signalled and not yet actioned it */
#endif /* SLAVE_SEM */

pid_t   parentpid = 0;                  /*  pid of
parent process                         */
pid_t   slave_formatter_pid = 0;        /*  pid of
slave_formatter process to write listbuf to file */
/*  fields relating to status of slave process as
reported by waitpid  */
int slave_wait_rc = 0;                  /*  rc
:  0 <=> active, pid <=> exited, -1 <=> error */
#ifdef _BSD
union wait slave_wait_status;           /*  status
:  meaningful only after exited       */
#else
int slave_wait_status;                  /*  status
:  meaningful only after exited       */
#endif
int     listbufshmid = 0;               /*  shmid of
listing output buffer memory  */
char   *listbufadr = 0;                 /* -> shm
area containing listing output buffer for queries to
write into followed by control fields */

volatile char   *listbufcur;            /* -> start
of current buffer for queries to write into */
volatile char   *listbufnxt;            /* -> next
byte to be written into in listing output buffer for
queries to write into */
volatile char *listbufcurhdp;           /*  remember
end of header */
volatile char *listbufcurtrp;           /*  remember
end of rows   */

struct listbufctlfields {               /*  listbuf
control fields */
   volatile char   *listbufprv;         /* -> next
byte to be transferred from listing output buffer to
file by slave_formatter */
   volatile char   *listbufhdp;         /* -> next
byte following header (prior to rows)
*/
   volatile char   *listbuftrp;         /* -> next
byte prior to trailer (following rows)
*/
   volatile char   *listbufend;         /* byte
following end of prev buffer */
   int              listbufqnum;        /* current
query number    (only needed for debugging and
errmsgs)     */
   int              wait_serializer;    /*
serialization for processes to wait: */
#define WAIT_NONE   0                   /* no
process waiting
*/
#define WAIT_PARENT 1                   /* parent
waiting for slave_formatter
*/
#define WAIT_SLAVE  2                   /*
slave_formatter waiting for parent
*/
#ifdef SLAVE_SEM
  int             parent_slave_semid;   /* semaphore
id of two semaphores on which parent and slave wait */
#define PARENT_SEM 0                    /* parent
waits on 0 and posts 1
*/
#undef SLAVE_SEM                        /* prepare
to define it with following value
*/
#define SLAVE_SEM 1                     /* slave
waits on 1 and posts 0
*/
#endif /* SLAVE_SEM */
   char             listbufname[256];   /*  filename
of output file          */
#ifdef TPCD_PROGRESS_FILE
   int listbufnextqnum_maybe;  /* next query number
(may not be reliable - valid only if != listbufqnum)
*/
   long   rows_fetch;                   /*  number
of rows returned */
   long sqlcode;                        /*  sqlcode
*/
   char               start_stamp[50];  /* start
time stamp for block  */
   char               end_stamp[50];    /* end time
stamp for block    */
#endif
 };
struct listbufctlfields *listbufctlp = 0; /*   -> the
control fields at the end of the shm area containing
buffer and control fields */
   /*  LISTBUFSIZE is size of each buffer - 1Gb should
be enough for largest query (Q11) */
   /*  since user may specify value, make sure it is a
multiple of a page in size        */
#ifndef LISTBUFSIZE
#define LISTBUFSIZE 0x0000000040000000L
#endif
   /*  now round up to a page boundary  */
#define RND_LISTBUFSIZE ((long)(((LISTBUFSIZE +
0x0FFFL)>>12)<<12))

  /*  the folllowing defines provide shorthand for the
control fields - note the preprocessor will not expand
recursively! */
#define listbufprv (listbufctlp->listbufprv)
#define listbufend (listbufctlp->listbufend)
#define listbufhdp (listbufctlp->listbufhdp)
#define listbuftrp (listbufctlp->listbuftrp)
#define listbufqnum (listbufctlp->listbufqnum)
#define wait_serializer (listbufctlp->wait_serializer)
```

```
#define listbufname (listbufctlp->listbufname)
#ifdef SLAVE_SEM
#define parent_slave_semid (listbufctlp-
>parent_slave_semid)
#endif /* SLAVE_SEM */

#ifdef _AIX
/*   functions to serialize and syncrhonize and update
atomically */
void gen_isync(void);         /*    synchronize the
order of my instruction pipeline */
#pragma mc_func gen_isync  {        \
  "4c00012c"                  /*         isync   */ \
}
void gen_sync(void);          /*    synchronize
processor caches                 */
#pragma mc_func gen_sync  {         \
  "7c0004ac"                  /*         sync   */ \
}
/*   compare_and_swap updates the 32-bit word from old
to new atomically
 *   and returns TRUE if successful, else FALSE
 */
#define compare_and_swap(_word_addr_, _old_val_,
_new_val_) (!_check_lock(_word_addr_, _old_val_,
_new_val_))
#elif defined(LINUX)
/*  must distinguish between ia32 / AMD which use lock
cmpxchg, and ia64 which requires use of compiler
intrinsic */
#if defined(_LINUX_IA64)
#include <asm/types.h>
       /*  from
/wsdb/db2_v82/daily/common/osse/core/inc/ossIA64Atomic
.h
       **
       **     OSSCSValue32 oldValue =
       **
(OSSCSValue32)_InterlockedCompareExchange_acq(
       **             (volatile
OSS_INTEL_ATOMIC32_CAS_INTRINSIC_TYPE_UNSIGNED
*)pAtomic,
       **             OSS_INTEL_ATOMIC32_CAS_CAST
newValue,
       **             OSS_INTEL_ATOMIC32_CAS_CAST
compareValue ) ;
       */

       /*   from linux-2.6.7/include/asm-
ia64/intel_intrin.h
       **
       **     __u64
_InterlockedCompareExchange_acq(volatile __u32 *dest,
__u64 xchg, __u64 comp);
       **
       **      typedef __signed__ int __s32;
       **      typedef unsigned int __u32;
       **
       **      typedef __signed__ long __s64;
       **      typedef unsigned long __u64;
       */

#ifdef _INTEL_COMPILER
/*  the following function is an intel compile built-
in intrinsic  */
#define cmpswap_ptr unsigned int *
#define cmpswap_val unsigned long
cmpswap_val _InterlockedCompareExchange_acq(volatile
cmpswap_ptr, cmpswap_val, cmpswap_val);
#else   /*  not _INTEL_COMPILER */
#include <asm/gcc_intrin.h>
#define cmpswap_ptr unsigned int *
#define cmpswap_val unsigned int
#define _InterlockedCompareExchange_acq(_ptr_, _new_,
_old_) ia64_cmpxchg4_acq(_ptr_, _new_, _old_)
#endif  /*  not _INTEL_COMPILER */
#else /*  not _LINUX_IA64 */
#define cmpswap_ptr int *
#define cmpswap_val int
#endif  /*  not _LINUX_IA64 */

static inline int compare_and_swap
(
   volatile int * const   pAtomic,
   const int              compareValue,
   const int              newValue
)
{
   cmpswap_val oldValue;

#if defined(_LINUX_IA64)

   oldValue =
(cmpswap_val)_InterlockedCompareExchange_acq
              (  (volatile cmpswap_ptr)pAtomic
              ,  (cmpswap_val)newValue
              ,  (cmpswap_val)compareValue );

#else /*  not _LINUX_IA64 */

/*   WARNING - the following presumes 32-bit intel
hardware.   needs alternative code for other hardware
types
**   source is taken from e.g. macro
OSS_IA32_CMPXCHG32 in
/wsdb/db2_v82/daily/common/osse/core/inc/ossIA32Atomic
.h
*/
   __asm__ __volatile__ ("lock cmpxchgl %2,%1\n\t"
     /* outputs */     : "=a" (oldValue),    /* %0 :
%eax */
                         "+m" (*pAtomic)     /* %1 :
memory location */
     /* inputs */      : "r" (newValue),     /* %2 :
register */
                         "0" (compareValue)  /* %3 ==
%0 == %eax */
     /* clobbers */    : "cc"                /*
condition registers (ZF) */
                         );

#endif  /*  not _LINUX_IA64 */

   return ( (unsigned int)oldValue == (unsigned
int)compareValue);
}
#endif   /*  LINUX */

#ifndef SLAVE_SEM
void handlSIGS(int sigsent,
                     int code, struct sigcontext
*handcontext)
{
  /*  we are invoked whenever signals 1 (HANGUP) or 15
(SIGTERM) or 30 (SIGUSR1) are issued  */
  int rc;

  signalled = sigsent;
     return;
}

#endif /* SLAVE_SEM */
#endif    /*  ifndef NO_PARALLEL_FORMAT_FETCH
parallel format-fetch only if requested */

#ifdef __cplusplus
inline
#endif
 int LPRINTF(FILE *stream, char *va_alist, ...)   /*
write query output to either buffer if it exists, else
directly to file */
{
#if defined(_HAVE_STDARG)  /*  for gcc on 64-bit linux
and some 32-bit linux we must use the stdarg macros as
the parm list is not in "natural" format */
#include <stdarg.h>
 va_list ap;
#else
 char *ap;
#endif
 char *format;
 int rc;

#if defined(_HAVE_STDARG)  /*  for gcc on 64-bit linux
and some 32-bit linux we must use the stdarg macros as
the parm list is not in "natural" format */
   va_start(ap ,va_alist); /*  initialise ap ->
beginning of arg list */
   format = va_alist;      /*  address second parm -
format string   */
#else
   ap = (char *) &va_alist;  /*  initialise ap ->
```

```
beginning of arg list */
   format = *((char **)ap);  /*  address second parm -
format string   */
   ap += sizeof(char *);     /*  advance parm pointer
-> following parm  */
#endif

#ifndef NO_PARALLEL_FORMAT_FETCH      /*   parallel
format-fetch only if requested */
   if (listbufadr)
   {
     /*  BIG WARNING   *  BIG WARNING   *  BIG WARNING
      *  We dont check for overwrite in here -
      *  as it is too expensive and difficult
      *  so ensure you set LISTBUFSIZE large enough to
hold output from largest query
      */
     rc = vsprintf((char *)listbufnxt, format, ap);
     listbufnxt += rc;
   }
   else
#endif  /*  NO_PARALLEL_FORMAT_FETCH      parallel
format-fetch only if requested */
     rc = vfprintf(stream, format, ap);

 return rc;
}

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 */
int slow_slave_count = 0;                 /* number of
times slave took longer to format prev than main did
to FETCH next */

/* 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

#ifndef TPCD_PLOAD_SCRIPTS
#ifdef SQLWINT
/*************************************************
*****************
 * Thread for parallel loading the UF orders data with
loading UF data *
 * for lineitem.
*
 *************************************************
****************/
DWORD WINAPI Load_UF_Data_Orders (LPVOID lpParm)
{
  int rc;
  DWORD dwRC;
  char *charptr;
#ifdef TPCD_PLOAD_API
  /*  use db2Load api - function source is in
tpcdUF.sqc  */
  if(verbose) {
     fprintf(stderr,"Calling Load_orders Update Pair
%d.\n", pupdatePair);
     fflush(stderr);
  }
  charptr=getenv("TPCD_FLATFILES");  /*  path for
input to load */
  rc =
(tpcd_load_staging((TPCDBATCH_INSERT+TPCDBATCH_ORDERS)
,dbname,pupdatePair,charptr,verbose));
  if(rc < 0) {
     fprintf(stderr,"Error %d Loading Orders UF
Data\n",rc);
     fflush(stderr);
  }
#else
    error "running db2 clp under Windows not yet
supported - code needed"
#endif
  Parallel_Load_Complete = TRUE;
  iParallel_Load_Result = rc;
  if (verbose) {
     fprintf(stderr, "UF_ORDERS_LOAD COMPLETE. rc =
%d\n", rc);
     fflush(stderr);
  }
  dwRC = (DWORD) rc;
  return(rc);
}
#endif /*  WINNT */
#endif /* not TPCD_PLOAD_SCRIPTS */


/*************************************************
*************
 * Converts packed decimal value to ascii string
*
 * The string is prefixed by - sign if negative and
not zero,      *
 * but not by + sign if positive, and leading zeros
are            *
 * suppressed.   The string is not null-terminated.
*
 * return length of ascii string.
 *************************************************
*************/
int dectoasc(
   char *decptr,
   char *asciiptr,
   short prec,
   short scal)
{
   /*   work left to right to facilitate removing
leading zeros  */
   int allzero = TRUE;
   char *srcptr;
   unsigned char sign;
   char *targptr, decimal_point = '.';
   int rc = 0;
   int tmpint;
   int left_nibble;   /*  1 if selecting left nibble,
0 if selecting right */
   int count, j, limit[2];

   targptr = asciiptr;
   srcptr = decptr;

   /* check validity of sign nibble and precision */
   if (((sign = sqlrx_get_right_nibble( *(decptr +
prec/2) )) < 0x0a)
       || (prec > SQL_MAXDECIMAL) || (prec < scal ))
   {
     rc = -1;
     goto exit;
   }/** end end if invalid sign value **/

   /*  if a negative sign, then insert - sign into
output.
    *  if the value is all zeros, then we will remove
this sign later
    */
   if (     (sign != SQLRX_PREFERRED_PLUS)
         && (sign != 0x0a)
         && (sign != 0x0e)
         && (sign != 0x0f)
       )
     *targptr++ = '-';

   limit[ 0 ] = prec - scal;       /*  number of
nibbles in integral part   */
   limit[ 1 ] = scal;              /*  number of
nibbles in fractional part */
   /*   since we work left to right,
    *   we must take care to start with the correct
```

```
leftmost digit.
    *   we cannot trust that an unwanted leftmost
digit is zero.
    */
   left_nibble = prec & 0x0001;    /*  start with left
if odd precision, else right */
   for( j = 0 ; j < 2 ; j++ )
   {
      for( count = limit[ j ] ; count > 0 ; count-- )
      {
         tmpint = ( left_nibble ?
                    sqlrx_get_left_nibble( *srcptr ) :
                    sqlrx_get_right_nibble(
*srcptr++ ) );
         if( tmpint > 9 )
         {
           rc = -2;
           goto exit;
         }

         if ( tmpint != 0 ) allzero = FALSE;
         if (!allzero)
            *targptr++ = (char)(tmpint + (int)'0');
         left_nibble = (1 - left_nibble);
      } /** end for loop over part of decimal */

      if( j == 0 )
         *targptr++ = decimal_point;
   } /** end loop over j           */

   rc = (targptr - asciiptr);
   if (allzero)   /*  if value is zero, simply return
zero regardless of sign */
   {
     *asciiptr = '0';
     rc = 1;
   }

   exit:
   if( rc < 0 )
   {
      fprintf (stderr,"decimal conversion has
failed\n");     fflush(stderr);
   }

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

#ifndef NO_PARALLEL_FORMAT_FETCH      /*   parallel
format-fetch only if requested */
void parent_is_exiting(void)
  /*   routine driven at parent exit to try to wake
slave in case he is still lingering.
   *   usually slave has already exited at this point
and there is nothing to do,
   *   so we simply do everything we can think of to
encourage slave to exit
   *   but don't check anything since usually they
will fail.
   */
{
#ifdef SLAVE_SEM
   struct sembuf parent_semaphore_operations;
   parent_semaphore_operations.sem_flg = 0; /*
operation flags */
#endif /* SLAVE_SEM */
   listbufend = 0;    /*   inform slave environment
has gone */
   compare_and_swap (&wait_serializer, WAIT_SLAVE,
WAIT_NONE);   /*  if he was waiting for me */
#ifndef SLAVE_SEM
   kill(slave_formatter_pid,SIGUSR1);     /*  post
slave */
#else /* SLAVE_SEM */
   parent_semaphore_operations.sem_num = SLAVE_SEM; /*
semaphore # SLAVE */
   parent_semaphore_operations.sem_op =  1; /*
semaphore operation POST */
   semop(parent_slave_semid,
&parent_semaphore_operations, 1);  /*  tell him to
start formatting   */
#endif /* SLAVE_SEM */
   return;
}

int  parent_rendezvous_point (struct global_struct
*g_struct, int proceed)
     /*    perform rendezvous for parent -
      *    tell slave to proceed if flag set,
      *    else tell slave to terminate
      */
{
   int rc = 0;
#ifdef SLAVE_SEM
   struct sembuf parent_semaphore_operations;
   parent_semaphore_operations.sem_flg = 0; /*
operation flags */
#endif /* SLAVE_SEM */
   parent_rendezvous_retry:
#ifdef _AIX
     gen_isync();           /*    order our
instruction pipeline */
#endif
     /*  we expect we will normally not have to wait
here - so first try to post slave       */
#ifndef SLAVE_SEM
     /*  note that slave's protocol ensures that our
signalled indicator cannot be set
      *       unless wait_serializer == WAIT_PARENT,
which cannot be the case at this point.
      */
#ifdef PARANOID
     if  (signalled)
/*  if already signalled     */
     {
       fprintf(stderr,"parent found myself signalled
when not waiting! wait_serializer= %d\n"
                      ,wait_serializer);
fflush(stderr);
       rc = 1;
     }
     else
#endif
#endif /* SLAVE_SEM */
#ifdef DEBUG_PARALLEL
       if (verbose)
       {
         fprintf(stderr,"parent_rendezvous_point
found wait_serializer= %d proceed=
%d\n",wait_serializer, proceed);
         fflush(stderr);
       }
#endif

     if  (compare_and_swap (&wait_serializer,
WAIT_SLAVE, WAIT_NONE)) /* slave was waiting for me */
     {
#ifdef DEBUG_PARALLEL
       if (verbose) {
         fprintf(stderr,"parent_rendezvous_point  set
wait_serializer= NONE\n");  fflush(stderr);
       }
#endif
       if (proceed)
       {
       listbufprv = listbufcur;        /*   start of
where to write  */
       listbufend = listbufnxt;        /*   beyond end
of where to write  */
       listbufhdp = listbufcurhdp;     /*   remember
end of header */
       listbuftrp = listbufcurtrp;     /*   remember
end of rows   */
       strcpy(listbufname,outstreamfilename);  /*
tell him the file name */
       listbufqnum = g_struct->qnum;   /*   tell him
the query number (only for debugging and msgs) */
#ifdef TPCD_PROGRESS_FILE
       listbufctlp->listbufnextqnum_maybe =
listbufqnum;  /*  initialise next qurynum == current
to indicate not yet valid */
       strcpy(listbufctlp->start_stamp,g_struct-
>s_info_ptr->start_stamp);
       strcpy(listbufctlp->end_stamp,g_struct-
>s_info_ptr->end_stamp);
       listbufctlp->rows_fetch = g_struct->s_info_ptr-
>rows_fetch;
       listbufctlp->sqlcode = g_struct->s_info_ptr-
>sqlcode;
#endif
       listbufnxt = listbufcur = (listbufadr +
RND_LISTBUFSIZE - (listbufcur - listbufadr));   /*
```

```
toggle to other buffer */
       listbufcurhdp = listbufnxt;     /*  initialise
listbufcurhdp to listbufnxt to indicate nothing (yet)
built in listbuf */
#ifdef _AIX
       gen_sync();           /*    sync
slave_formatter's cache */
#endif
       }
       else
       {
         listbufprv = 0;           /*  tell slave to
terminate  */
         listbufqnum = 999999999;  /*  indicate end of
queries in following messages */
       }
#ifndef SLAVE_SEM
       rc = kill(slave_formatter_pid,SIGUSR1);
/*  tell him to start formatting   */
#else /* SLAVE_SEM */
       parent_semaphore_operations.sem_num =
SLAVE_SEM; /* semaphore # SLAVE */
       parent_semaphore_operations.sem_op =  1; /*
semaphore operation POST */
       rc = semop(parent_slave_semid,
&parent_semaphore_operations, 1);  /*  tell him to
start formatting   */
#endif /* SLAVE_SEM */
       if (rc < 0) {
         fprintf(stderr,"\nmain rc %d errno %d from
trying to post slave_formatter pid %d for query %d\n"
           ,rc, errno,
slave_formatter_pid,listbufqnum); fflush(stderr);
       }
       else if (verbose) {
         fprintf(stderr,"\nmain posted slave_formatter
for query %d\n" ,listbufqnum);  fflush(stderr);
       }
     }
     else
     {
       if (compare_and_swap (&wait_serializer,
WAIT_NONE, WAIT_PARENT)) /* slave not waiting for me
*/
       {
#ifdef DEBUG_PARALLEL
         if (verbose) {
           fprintf(stderr,"parent_rendezvous_point
set wait_serializer= %d
WAIT_PARENT\n",wait_serializer);  fflush(stderr);
         }
#endif
          /*  it's possible slave may have terminated,
          **  so to avoid hanging, check here
          **  unfortunately kill ,,,0 returns 0 for a
defunct process - need to check waitpid as well
          **  NOTE re return values waitpid -  (there
are two - rc and status)
          **       when WNOHANG is specified:
          **          rc is never < 0 unless invalid
parm of some sort
          **          rc == 0     <==>    no child was
available
          **          rc >  0     <==>    rc = pid of
child which has exited (the case in which we are
interested)
          **                                  and in this
case, status is meaningful
          */
         if (       ((rc =
kill(slave_formatter_pid,0)) < 0)
             ||    (slave_wait_rc ==
slave_formatter_pid)  /*  never call waitpid again
after prev call reported pid */
             ||    (    (    (slave_wait_rc =
waitpid(slave_formatter_pid,&slave_wait_status,WNOHANG
))
#ifdef DEBUG_SLAVE_EXISTENCE
                          ,
( fprintf(stderr,"slave_formatter %d waitpid rc %d
status %X\n"
                               ,slave_formatter_pid
,slave_wait_rc ,slave_wait_status)
                             ) , fflush(stderr) ,
pause()
#endif
                             ,  (slave_wait_rc ==
slave_formatter_pid)  /*  waitpid is returning
information about slave pid */
                          )
                    /*   we used to check for
(WIFEXITED(slave_wait_status) here
                    **   but it returns true only if the
slave exited normally (no signal, no error rc)
                    **   and we don't care what status
the slave exited with
                       &&
(WIFEXITED(slave_wait_status))         **  slave has
exited normally  **
                    */
                      )
             )
             goto slave_died;
         if (verbose) {
           fprintf(stderr,"slow slave_formatter on
query %d\n",g_struct->qnum); fflush(stderr);
         }
         slow_slave_count++;
#ifdef DEBUG_PARALLEL
         if (verbose) {
           fprintf(stderr,"parent about to wait,
wait_serializer= %d\n",wait_serializer);
fflush(stderr);
         }
#endif
#ifndef SLAVE_SEM
         sigprocmask(SIG_BLOCK,&blockusermask,&tempmas
k); /*  temporarily block signal USR1... */
         while (signalled != SIGUSR1)              /*
check we have received the correct signal */
           sigsuspend(&tempmask);                  /*
wait for signal from slave_formatter  */
         sigprocmask(SIG_SETMASK,&tempmask,0);     /*
normal mask allows USR1 */
         signalled = 0;                            /*
reset indicator       */
#else /* SLAVE_SEM */
  wait_for_signal:                 /*   wait till he
signals me */
         parent_semaphore_operations.sem_num =
PARENT_SEM; /* semaphore # PARENT */
         parent_semaphore_operations.sem_op = -1; /*
semaphore operation WAIT */
         rc = semop(parent_slave_semid,
&parent_semaphore_operations, 1);  /*  wait for signal
from slave_formatter */
         if (rc < 0)
         {
           if (errno == EINTR)               /*
interrupted - maybe someone debugging me   */
             goto wait_for_signal;           /*  so
just go round again                */
           if (errno != EIDRM) {             /*  not
just parent terminating */
             fprintf(stderr,"\nmain rc %d errno %d
from trying to wait for slave_formatter pid %d for
query %d\n"
                    ,rc, errno,
slave_formatter_pid,listbufqnum); fflush(stderr);
           }
         }
#endif /* SLAVE_SEM */
#ifdef _AIX
         gen_isync();           /*    order our
instruction pipeline */
#endif
       }
       /*  we have not yet set up the control fields
to tell the slave what to do ... so ... */
       goto parent_rendezvous_retry;
     }
 return_from_func:
     return rc;
       slave_died:
         fprintf(stderr,"slave_formatter %d has
exited\n",slave_formatter_pid); fflush(stderr);
     return -1;
}
#endif  /*  NO_PARALLEL_FORMAT_FETCH      parallel
format-fetch only if requested */

/*************************************************
```

```
*************/
/* 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
#ifdef TPCD_PROGRESS_FILE
                          ,0         /*  sqlcode
*/
                          ,0         /*  number of rows
returned    */
#endif
                          ,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, 0,
    NULL, "\0", "\0", "\0", NULL, NULL, NULL, 0 };

#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, 0,
    NULL, "\0", "\0", "\0", NULL, NULL, NULL, 0 };
#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"                     *
   *
*
   *************************************************
************************/
#ifndef NO_PARALLEL_FORMAT_FETCH      /*   parallel
format-fetch only if requested */
   parentpid = getpid();              /*   my pid  */
   /*  NOTE   we never use parallel format-fetch if
the tpcdbatch program is told to run updates,
   **         either with or without queries.
   **         The usual way to run updates is in a
separate tpcdbatch step which runs updates only,
   **         and for this setup, parallel format-
fetch can be used in the tpcdbatch which runs the
queries.
   **         i.e. - in terms of tpcdbatch parameters
-
   **         parallel format-fetch only when the -u/-
U flag is set to P1 or T1
   */
   if (g_struct.c_l_opt->update < 2)  /*   no parallel
format-fetch if running updates */
   {
#ifndef SLAVE_SEM
      /*   set up the signal handling function  */
      sigemptyset(&emptymask);
      sigemptyset(&blockusermask);
      sigaddset(&blockusermask, SIGUSR1);
      sigprocmask(SIG_SETMASK,&emptymask,0);       /*
normal mask allows USR1 */

      chsigaction.sa_handler =
(void(*)(int))&handlSIGS;
      chsigaction.sa_mask = emptymask;
      chsigaction.sa_flags = 0;
      sigaction(SIGUSR1, &chsigaction, &olsigaction);

#endif /* SLAVE_SEM */
     /*   get a shm area of size 2*bufsize plus area
for control fields */
    if (   ( (listbufshmid =
shmget(IPC_PRIVATE,((RND_LISTBUFSIZE<<1) +
sizeof(struct listbufctlfields)),(IPC_CREAT|S_IRUSR|
S_IWUSR))) == -1)
         ||
           ( (listbufadr = (char *)(shmat
(listbufshmid, (void *)0, 0))) == (char *)(-1))
       )
     {
       fprintf(stderr,"unable to acquire shared-mem
listbuf, shmid %d address %p reason
%d\n",listbufshmid,listbufadr,errno);
       fflush(stderr);
       listbufadr = 0;
       return errno;
     }

     {
       pid_t forkrc;

       listbufctlp = (struct listbufctlfields
*)(listbufadr+(RND_LISTBUFSIZE<<1)); /* locate control
fields at the end of the buffer */
       if (verbose) {
         fprintf(stderr,"LISTBUFSIZE macro value
0x%pL, rounded to 0x%pL, acquired shared-mem listbuf,
shmid %d address %p size %lld\n"
                 ,LISTBUFSIZE,RND_LISTBUFSIZE,listbufshm
id,listbufadr ,((RND_LISTBUFSIZE<<1) + sizeof(struct
listbufctlfields))); fflush(stderr);
       }
       listbufnxt = listbufcur = listbufadr;
       listbufprv = listbufend = 0;
#ifndef SLAVE_SEM
       signalled = 0;                          /*
initially no process has been signalled */
#else /* SLAVE_SEM */

       if ((parent_slave_semid =
semget(IPC_PRIVATE,2,IPC_CREAT|IPC_EXCL|S_IRUSR|
S_IWUSR|S_IRGRP|S_IWGRP)) < 0)
       {
         fprintf(stderr,"unable to acquire semaphores
errno %d\n",errno); fflush(stderr);
         goto rmshm;
       }
#endif /* SLAVE_SEM */
       wait_serializer = WAIT_NONE;          /*
initially no process waiting            */
       if (forkrc = fork())
```

```
       {
        /*  I am parent - continue  */
        slave_formatter_pid = forkrc;
        if (verbose) {
          fprintf(stderr,"forked slave_formatter pid
%d\n",slave_formatter_pid); fflush(stderr);
        }
        atexit (parent_is_exiting);

      }
       else   /*  child - wait for instructions */
      {
#ifdef TPCD_PROGRESS_FILE
        /*  append stream number to specified name */
#define SIZEOF_TPCD_PROGRESS_FILE
sizeof(TPCD_PROGRESS_FILE)
        char tpcd_progress_file_name[
SIZEOF_TPCD_PROGRESS_FILE + 16 ];
#endif
        nice (10);   /*  lower my priority by 10
notches to prevent delaying parent */
#ifdef TPCD_PROGRESS_FILE
        strcpy (tpcd_progress_file_name,
TPCD_PROGRESS_FILE );
        sprintf (tpcd_progress_file_name +
SIZEOF_TPCD_PROGRESS_FILE - 1, ".%d",
c_l_opt.intStreamNum ); /* -1 to overwrite null */
        progress_file =
fopen(tpcd_progress_file_name,"a");
        if (!progress_file) {
          fprintf(stderr,"error %d from trying to
open progress file\n",errno);  fflush(stderr);
        }
#endif
        goto slave_formatter_wait;
      }
     }
   }
#endif   /*  ifndef NO_PARALLEL_FORMAT_FETCH
parallel format-fetch only if not windows */

  /************************************************
******************/
  /* 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 */
#ifndef NO_PARALLEL_FORMAT_FETCH  /*   parallel
format-fetch only if requested */
#ifdef TPCD_PROGRESS_FILE
      /*    store querynum of next query in shared mem
so slave can annotate progress file.
      **    Note this is not synchronised with slave
activity as that is too expensive -
      **          if he has already gone into wait, he
will miss it.
      */
      if (listbufctlp)   /*  make sure we have the
shared mem area */
        listbufctlp->listbufnextqnum_maybe =
g_struct.qnum;
#endif
#endif
    }
    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_ti
me ));  /* 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 */

      LPRINTF(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)
          {
            LPRINTF(outstream,"Using default qgen seed
file");
          }
        else
          LPRINTF(outstream,"Seed used =
%ld",g_struct.lSeed);

        LPRINTF(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"    *
   *
*
   *************************************************
************************/
#ifndef NO_PARALLEL_FORMAT_FETCH      /*   parallel
format-fetch only if requested */
   if (slave_formatter_pid)       /*  if our
slave_formatter is writing the output files */
   {
     /*********************************************
**************/
     /*    rendezvous
*/
     /*    ensure slave has finished formatting
previous query      */
     /*    before proceeding to terminate.
*/
     /*********************************************
**************/
     parent_rendezvous_point(&g_struct,0); /*
rendezvous and tell slave to terminate */
     /*  at this point, if we have not yet received
positive confirmation that slave has terminated,
     **  then wait for slave to terminate to avoid
slave hitting problems with IPC's disappearing under
him,
     **  and also to collect slave status in order to
remove defunct (zombie) process.
     **  NOTE that there is a small possibility that
we hang here since we wait -
     **       however this can happen only if the
slave also hung, in which case a parent hang
     **       does not make it any worse.
     **  WHEREAS if we do not wait here, slave could
segv if very slow to detect our instruction to
terminate.
     */
     if (slave_wait_rc == 0)             /*  waitpid
has not yet reported pid */
     {
       if (verbose) {
         fprintf(stderr,"parent %d about to wait for
slave %d to exit\n",parentpid, slave_formatter_pid);
fflush(stderr);
       }
       slave_wait_rc = waitpid(slave_formatter_pid,
&slave_wait_status,0);
     }
     if (verbose) {
       fprintf(stderr,"parent %d about to terminate,
slow_slave_count %d\n",parentpid, slow_slave_count);
fflush(stderr);
     }
   }
#ifdef SLAVE_SEM
   semctl(parent_slave_semid,2,IPC_RMID);  /* remove
the semaphore set */
#endif /* SLAVE_SEM */

 rmshm:
   if (listbufshmid)
     shmctl(listbufshmid, IPC_RMID,0);     /* remove
the shared-mem listing buffer */
#endif  /*  NO_PARALLEL_FORMAT_FETCH      parallel
format-fetch only if requested */


  return(0);

   /*************************************************
*************/
   /*   slave_formatter processing
*/
   /*************************************************
*************/
#ifndef NO_PARALLEL_FORMAT_FETCH      /*   parallel
format-fetch only if requested */
   slave_formatter_wait:   /*  slave_formatter - wait
for instructions */
   {
     long size;
     FILE *workstream;
     int   workfiled;
#ifdef SLAVE_SEM
     struct sembuf slave_semaphore_operations;
#endif /* SLAVE_SEM */
  int waitstatus;                     /*   wait status
of the two processes */

     /*********************************************
**************/
     /*    rendezvous
*/
     /*    ensure parent has reached end of FETCHing
query          */
     /*    before proceeding to format it
*/
     /*********************************************
**************/
#ifdef SLAVE_SEM
   slave_semaphore_operations.sem_flg = 0; /*
operation flags */
#endif /* SLAVE_SEM */
   slave_rendezvous_retry:
#ifdef _AIX
     gen_isync();           /*    order our
instruction pipeline */
#endif
#ifdef DEBUG_PARALLEL
     if (verbose) {
       fprintf(stderr,"slave  found wait_serializer=
%d\n",wait_serializer); fflush(stderr);
     }
#endif
     /*  we expect we will normally have to wait here
- so first try to put myself into wait */
#ifndef SLAVE_SEM
     /*  note that parent's protocol ensures that our
signalled indicator cannot be set
      *       unless wait_serializer == WAIT_SLAVE,
which cannot be the case at this point.
      */

#ifdef PARANOID
     if  (signalled) {
/*  if already signalled     */
       fprintf(stderr,"slave_formatter found myself
signalled when not waiting! wait_serializer= %d\n"
                      ,wait_serializer);
fflush(stderr);
     }
     else
#endif
#endif /* SLAVE_SEM */
       if (compare_and_swap (&wait_serializer,
WAIT_NONE, WAIT_SLAVE)) /* parent not waiting for me
*/
       {
#ifdef DEBUG_PARALLEL
         if (verbose) {
           fprintf(stderr,"slave  set wait_serializer=
%d WAIT_SLAVE\n",wait_serializer); fflush(stderr);
         }
#endif
#ifdef SLAVE_SEM
         if (verbose) {
           fprintf(stderr,"slave_formatter about to
suspend after query %d listbufsize %lld\n",
listbufqnum, (long long)(listbufend-listbufprv));
           fflush(stderr);
         }
#endif /* SLAVE_SEM */
  wait_for_signal:                /*   wait till he
signals me */
#ifdef DEBUG_PARALLEL
         if (verbose) {
           fprintf(stderr,"slave  about to wait,
wait_serializer= %d\n",wait_serializer);
fflush(stderr);
         }
#endif
```

```
#ifndef SLAVE_SEM
        /*  block USR1 before checking once more so
that if not arrived yet, we will be awoken */
        sigprocmask(SIG_BLOCK,&blockusermask,&prevmas
k);  /*  temporarily block signal USR1... */
        if (!signalled)
/*  if not already signalled     */
#else /* SLAVE_SEM */
        slave_semaphore_operations.sem_num =
SLAVE_SEM; /* semaphore # SLAVE */
        slave_semaphore_operations.sem_op = -1; /*
semaphore operation WAIT */
        if ((semop(parent_slave_semid,
&slave_semaphore_operations, 1)) < 0)
#endif /* SLAVE_SEM */
        {
#ifndef SLAVE_SEM
          if (verbose) {
            fprintf(stderr,"slave_formatter about to
suspend after query %d listbufsize %lld\n",
listbufqnum, (long long)(listbufend-listbufprv));
            fflush(stderr);
          }
          do
          {
            sigsuspend(&prevmask);
/*   wait for signal from parent */
          } while (signalled != SIGUSR1);         /*
check we have received the correct signal */
#else /* SLAVE_SEM */
          if (errno == EINTR)                /*
interrupted - maybe someone debugging me . */
            goto wait_for_signal;            /*  so
just go round again                      */
          if (errno != EIDRM) {              /*  not
just parent terminating */
            fprintf(stderr,"slave_formatter error %d
from semop wait \n", errno); fflush(stderr);
          }
          goto end_slave;         /*  I should
terminate on any error other than interrupt */
#endif /* SLAVE_SEM */
        }
#ifndef SLAVE_SEM
        sigprocmask(SIG_SETMASK,&prevmask,0);     /*
normal mask allows USR1 */
        signalled = 0;                            /*
reset indicator        */
#endif /* SLAVE_SEM */
      }
      else
        /*  note that if parent was waiting for me,
         *       then after posting parent, I must
wait for him to set up my control fields
         */
      if  (compare_and_swap (&wait_serializer,
WAIT_PARENT, WAIT_SLAVE)) /* parent was waiting for me
*/
      {                                           /*  by
setting wait_serializer to WAIT_SLAVE, */
                                                  /*  we
permit parent to signal me             */
#ifdef DEBUG_PARALLEL
        if (verbose) {
          fprintf(stderr,"slave  set wait_serializer=
%d WAIT_SLAVE\n",wait_serializer); fflush(stderr);
        }
#endif
#ifndef SLAVE_SEM
        signalled = 0;                            /*
reset indicator to ensure I don't get false value  */
        kill(parentpid,SIGUSR1);                  /*
then tell him to resume   */
#else /* SLAVE_SEM */
        slave_semaphore_operations.sem_num =
PARENT_SEM; /* semaphore # PARENT */
        slave_semaphore_operations.sem_op =  1; /*
semaphore operation POST */
        if ((semop(parent_slave_semid,
&slave_semaphore_operations, 1)) < 0)
        {
          fprintf(stderr,"slave_formatter error %d
from semop post \n", errno);  fflush(stderr);
          goto end_slave;
        }
#endif /* SLAVE_SEM */
          goto wait_for_signal;                   /*  and
wait till he signals me */
        }
        else goto slave_rendezvous_retry;

        /*  one way or another, we can proceed with
formatting this query  */
#ifdef DEBUG_PARALLEL
      if (verbose) {
        fprintf(stderr,"slave_formatter found
listbufprv %p listbufend %p for query
%d\n",listbufprv, listbufend, listbufqnum);
        fflush(stderr);
      }
#endif
#ifdef _AIX
        gen_isync();            /*    order our
instruction pipeline */
#endif
        size = (listbufhdp - listbufprv);
#ifdef DEBUG_SLAVE_EXISTENCE
   debug_wait:
        fprintf(stderr,"slave waiting to be debugged
after query %d\n", listbufqnum);
        fflush(stderr);
        pause();
        goto debug_wait;
#endif
        if (listbufprv && listbufend)
        {
          if (size == 0)
          {
            if (verbose) {
              fprintf(stderr,"slave_formatter, nothing
to format for query %d\n", listbufqnum);
fflush(stderr);
            }
            goto slave_rendezvous_retry;
          }
#ifdef TPCD_PROGRESS_FILE
          if (progress_file)
          {
              fprintf(progress_file,"q%2d started %s
completed %s sqlcode %5d rowcount %8ld"
#ifdef TPCD_PROGRESS_LISTBUF
                                      " listbufsize
%lld"
#endif
                                      "\n"
                          ,listbufqnum, listbufctlp-
>start_stamp, listbufctlp->end_stamp, listbufctlp-
>sqlcode, listbufctlp->rows_fetch
#ifdef TPCD_PROGRESS_LISTBUF
                                      ,(long
long)(listbufend-listbufprv)
#endif
                        );
              fflush(progress_file);
          }
#endif
          if (workstream = fopen(listbufname,
WRITEMODE))
          {
            /*  first, write the lines which precede
the rows */
            if ((fwrite ((void *)listbufprv, 1, size,
workstream)) != size)  { /*  write the buffer */
              fprintf(stderr,"error on writing %d
bytes to outstream errno %d\n",size, errno);
fflush(stderr);
            }
#ifdef DEBUG_PARALLEL
            else if (verbose) {
              fprintf(stderr,"wrote %d bytes to
outstream %s for query %d\n",size, listbufname,
listbufqnum);  fflush(stderr);
            }
#endif
            fflush(workstream);
            /*  next, format the rows */
            format_saved_rows(workstream, listbufhdp,
listbuftrp);
            size = (listbufend - listbuftrp);
            /*  last, write the lines which follow the
rows */
            if ((fwrite ((void *)listbuftrp, 1, size,
```

```
workstream)) != size) {  /*  write the buffer */
              fprintf(stderr,"error on writing %d
bytes to outstream errno %d\n",size, errno);
fflush(stderr);
            }
#ifdef DEBUG_PARALLEL
           else if (verbose) {
             fprintf(stderr,"wrote %d bytes to
outstream %s for query %d\n",size, listbufname,
listbufqnum);  fflush(stderr);
            }
#endif
            fclose(workstream);
#ifdef _AIX
            gen_sync();                                  /*
sync parent's cache */
#endif
#ifdef TPCD_PROGRESS_FILE
          /*  special request from Kwai -wants to see
qnum of next query now being fetched by parent -
          **  this is a not-completely-reliable best-
effort.
          **  parent initialises listbufnextqnum_maybe
to previous query, then resets it to next one as soon
as he knows it.
          **  IF he does that BEFORE we reach here,
then we find it and report it.
          **  Note that we can NEVER report in-
progress for first query of stream (sorry)
          */
          if (  (progress_file) && ( listbufctlp-
>listbufnextqnum_maybe != 0 ) && ( listbufctlp-
>listbufnextqnum_maybe != listbufqnum ))
          {
              fprintf(progress_file,"q%2d in
progress\n", listbufctlp->listbufnextqnum_maybe);
              fflush(progress_file);
          }
#endif
            goto slave_rendezvous_retry;
          }
          else {
            fprintf(stderr,"slave_formatter error
errno %d on opening %s\n",errno,listbufname);
fflush(stderr);
          }
        }
        else if (verbose) {
          fprintf(stderr,"slave_formatter terminating,
listbufprv= %p listbufend= %p size=
%ld\n",listbufprv,listbufend,size);
          fflush(stderr);
        }
#ifdef _AIX
        gen_sync();                                  /*
sync parent's cache */
#endif
   end_slave:
#ifdef TPCD_PROGRESS_FILE
        if (progress_file)
           fclose(progress_file);
#endif
     /*  try to force state of WAIT_NONE to prevent
parent from hanging  */
     do {
       if (waitstatus = wait_serializer)    /*    a
process is waiting */
        {
#ifdef DEBUG_PARALLEL
         if (verbose) {
           fprintf(stderr,"slave  found
wait_serializer= %d\n",wait_serializer);
fflush(stderr);
         }
#endif
          if  (waitstatus = compare_and_swap
(&wait_serializer, waitstatus,  WAIT_NONE))   /*
successfully reset */
          {
#ifdef DEBUG_PARALLEL
            if (verbose) {
              fprintf(stderr,"slave  set
wait_serializer= 0 NONE\n");  fflush(stderr);
            }
#endif
#ifndef SLAVE_SEM
            kill(parentpid,SIGUSR1);
/*  then tell parent to resume   */
#else /* SLAVE_SEM */
            slave_semaphore_operations.sem_num =
PARENT_SEM; /* semaphore # PARENT */
            slave_semaphore_operations.sem_op =  1;
/* semaphore operation POST */
            if ((semop(parent_slave_semid,
&slave_semaphore_operations, 1)) < 0)
            {
              if (kill(parentpid,0) == 0) {
                fprintf(stderr,"slave_formatter error
%d from semop post \n", errno); fflush(stderr);
              }
            }
#endif /* SLAVE_SEM */
          }
        }
        else goto return_slave;              /*  then
tell him to resume   */
      }  while (waitstatus == 0);            /*
continue while failed to reset wait_serializer */
   return_slave:
     return 0;
   }
#endif  /*  NO_PARALLEL_FORMAT_FETCH      parallel
format-fetch only if requested */

}  /* 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) {

  char *charptr;

  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"));

   if (charptr = getenv("TPCD_INTER_UF_CHILD_DELAY"))
         inter_uf_child_delay = atof(charptr);
   else  inter_uf_child_delay = 0.0;

   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");
   LPRINTF(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));
            LPRINTF(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);
            LPRINTF(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);
            LPRINTF(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*/
            LPRINTF(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;
         }

         LPRINTF(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);

   LPRINTF(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 */
      if (select_status=strtok(temp_str," ")) /*
assignment intentional - verify there exists a non-
blank */
      {
        if (    (stmt_str.data[0] == '(') ||
(!strcmp(select_status,"SELECT"))
             || (!strcmp(select_status,"VALUES"))
             || (!strcmp(select_status,"WITH"))
           )
          return TPCDBATCH_SELECT;
        else if (*select_status == '.')   /*
optimizer driver command */
          return TPCDBATCH_OPT_DRIVER;
      }
      /*   if we reach here, either no non-blank char
or not a SELECT or opt driver */
      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  */
}


#ifndef NO_PARALLEL_FORMAT_FETCH      /*   parallel
format-fetch only if requested */

int format_saved_rows(FILE *formstream, char
*formstart, char *formend)
/*    format sqlda values stored in buffer between
formstart, formend */
{
   int rc = 1;
   char *charptr;              /*  to march through
the buffer            */
   int *col_lengths;           /*  the tpcdbatch
column formatting lengths */
   int nvars_per_row;          /*  from the sqlda.sqld
*/
   int    col;                          /* Column
counter             */
   int    col_type;                     /* Type of
column              */
   int    col_type_ign_null;            /* Type of
column ignoring nullability   */
   int    saved_length;                 /* byte-
length of saved value            */
   short  *sav_lengthptr;               /*  ->
saved length field */
   int    format_len;           /*  the tpcdbatch
column formatting length  */
   double float_val;
   short           m,n;        /* precision and scale
for decimal conversion    */
   char   output_line[384];     /*  assemble all
printed values into this prior to writing */
   char  *output_ptr;           /* -> next char to be
stored in output_line  */
   int    line_len;             /*  length of data
stored in output_line     */

   charptr = formstart;
   nvars_per_row = (int)(*(short *)charptr); charptr
+= 2;  /*   retrieve the sqld and advance  */
  /*   retrieve the tpcdbatch column formatting
lengths and advance  */
   col_lengths = (int *)charptr; charptr +=
(nvars_per_row * sizeof(col_lengths[0]));
#ifdef DEBUG_PARALLEL
   if (verbose)
     fprintf(stderr,"format_saved_rows, buffer
addresses from %p to %p, nvars_per_row= %d\n"
              ,formstart ,formend ,nvars_per_row);
#endif
   while (charptr < formend)
   {
     output_ptr = output_line;          /* -> next
char to be stored in output_line  */
     for (col=0; col<nvars_per_row; col++)    /* loop
through columns  */
     {
       col_type = (int)(*(short *)charptr); charptr +=
2;  /*   retrieve the col type and advance  */
       sav_lengthptr = (short *)charptr;
/*  ->  saved length field */
       saved_length = (int)(*(short *)charptr);
charptr += 2;  /*   retrieve the col leng and advance
*/
       format_len = col_lengths[col];
/*  the tpcdbatch column formatting length  */
       if (saved_length == -1)
/*  indicates a null  */
         output_ptr += sprintf(output_ptr, "%* n/a
",(format_len-3));
       else
       {
         col_type_ign_null = col_type & 0xFFFFFFFE; /*
turn off the nullable indicator as we are not
interested in it from here on */
         switch (col_type_ign_null)
         {
          case SQL_TYP_SMALL:
            output_ptr += sprintf(output_ptr, "%*hd
",format_len, *(short *)charptr);
            break;

          case SQL_TYP_INTEGER:
            output_ptr += sprintf(output_ptr, "%*ld
",format_len, *(sqlint32 *)charptr);
            break;

          case SQL_TYP_BIGINT:
            output_ptr += sprintf(output_ptr, "%*lld
",format_len, *(sqlint64 *)charptr);
            break;

          /*   for all of the following, save_sqlda
appended a null terminator */
```

```
          case SQL_TYP_CHAR:
          case SQL_TYP_VARCHAR:
          case SQL_TYP_LONG:
          case SQL_TYP_CSTR:
          case SQL_TYP_DATE:
          case SQL_TYP_TIME:
          case SQL_TYP_STAMP:
            memcpy(output_ptr, charptr, saved_length);
            output_ptr += saved_length;
            if ((rc = (format_len + 2 - saved_length))
> 0) /* +2 for spaces after value */
            {
              memset(output_ptr, ' ', rc); output_ptr
+= rc;
            }
            break;

          case SQL_TYP_FLOAT:
            if (saved_length == 8)
              float_val = *(double *)charptr;
            else
            float_val = (double)(*(float *)charptr);
            if ( fabs(float_val) <
TPCDBATCH_PRINT_FLOAT_MAX  )
                output_ptr += sprintf(output_ptr,
"%#*.3f  ",format_len, float_val);
              else
                output_ptr += sprintf(output_ptr, "%*e
",format_len,
                      float_val);
            break;

          case SQL_TYP_DECIMAL:
            m=((short)(*(struct declen
*)sav_lengthptr).m);
            n=((short)(*(struct declen
*)sav_lengthptr).n);
            /*  recalculate saved_length to be the
number of bytes saved (to be skipped over)  */
            saved_length = (m + 1)/2;
            /*  dectoasc converts the packed decimal
value to ascii string
             *   and returns the length of the string
             */
            if ((rc =
dectoasc(charptr,output_ptr,m,n)) <= 0)
            {
              fprintf(stderr, "\nThe decimal value
could not be converted.\n");
              return rc;
            }
            output_ptr += rc;
            if ((rc = (format_len + 2 - rc)) > 0) /*
+2 for spaces after value */
            {
              memset(output_ptr, ' ', rc);  output_ptr
+= rc;
            }
            break;

          default:
            fprintf(stderr,"format_saved_rows: Unknown
column type (%d)\n",col_type);
            break;
         }
         charptr += saved_length;   /*  advance
through saved_data buffer  */
       }   /*  end not a null column     */
     }     /*  end loop through columns  */
     *output_ptr = '\n';    /*  append linefeed  */
     line_len = (output_ptr - output_line +1);
/*  length of line to be written including linefeed
*/
#ifdef CHECK_LINE_OVERWRITE
     if (line_len > sizeof(output_line))
     {
       fprintf(stderr,"OOOPPPSSS   slave wrote %d
bytes into report line buffer of size %d, increase
it\n",line_len,sizeof(output_line));
       fflush(stderr);
       return -1;
     }
#endif
     if ((rc = (fwrite ((void *)output_line, 1,
line_len, formstream))) != line_len) {  /*  write the
buffer */
       fprintf(stderr,"error on writing %d bytes to
outstream rc %d errno %d\n",line_len, rc, errno);
fflush(stderr);
     }
#ifdef DEBUG_PARALLEL
     else if (verbose) {
       fprintf(stderr,"wrote %d bytes to
outstream\n",line_len);  fflush(stderr);
     }
#endif
   }       /*  end loop through listbuf  */
   return rc;
}

void save_sqlda(struct sqlda *sqlda)
          /*  save sqlda row to listbuf for later
formatting              */
{         /*  for each column, save type||leng||data
*/
   int    col;                              /* Column
counter              */
   short  col_type;                         /* Type of
column             */
   short  col_type_ign_null;                /* Type of
column ignoring nullability   */
   short  saved_length;                     /*  byte-
length of saved value           */
   short  *sav_lengthptr;                   /*  ->
saved length field */
   char   *col_dataptr;

#if defined(DEBUG_PARALLEL) &&
defined(DEBUG_SAVE_SQLDA)
   if (verbose)
     fprintf(stderr,"save_sqlda: listbufnxt %p
ncolumns= %d\n",listbufnxt,sqlda->sqld);
#endif
   for (col=0; col<sqlda->sqld; col++)    /* Loop
through column count  */
   {
      col_type=sqlda->sqlvar[col].sqltype;
      *((short *)listbufnxt) = col_type; listbufnxt +=
2;     /*  save type and advance */
      if (    (col_type & 0x0001)               /*
coltype is odd */
           && (*(sqlda->sqlvar[col].sqlind))      /*
ind is set     */
         )
      *((short *)listbufnxt) = -1;              /*
indicate n/a (null)  */
      else
      {
        col_type_ign_null = col_type & 0xFFFE;   /*
turn off the nullable indicator as we are not
interested in it from here on */
        saved_length=sqlda->sqlvar[col].sqllen;  /*
tentatively - may be adjusted depending on specific
type */
        col_dataptr = sqlda->sqlvar[col].sqldata;
        sav_lengthptr = (short *)listbufnxt;
/*  ->  saved length field */
        listbufnxt += 2;
/* advance to -> data      */

         switch (col_type_ign_null)
         {
          case SQL_TYP_SMALL:
            *sav_lengthptr = saved_length;
/*  save length */
            *((sqlint16 *)listbufnxt) = *((sqlint16
*)(sqlda->sqlvar[col].sqldata)); listbufnxt += 2;
/*  save data and advance */
            break;

          case SQL_TYP_INTEGER:
            *sav_lengthptr = saved_length;
/*  save length */
            *((sqlint32 *)listbufnxt) = *((sqlint32
*)(sqlda->sqlvar[col].sqldata)); listbufnxt += 4;
/*  save data and advance */
            break;

          case SQL_TYP_BIGINT:
            *sav_lengthptr = saved_length;
/*  save length */
            *((sqlint64 *)listbufnxt) = *((sqlint64
```

```
*)(sqlda->sqlvar[col].sqldata)); listbufnxt += 8;
/*  save data and advance */
            break;

          case SQL_TYP_FLOAT:
            *sav_lengthptr = saved_length;
/*  save length */
            if (saved_length == 8)
                *((sqlint64 *)listbufnxt) =
*(sqlint64 *)(sqlda->sqlvar[col].sqldata);   /*  save
data and advance */
            else *((sqlint32 *)listbufnxt) =
*(sqlint32 *)(sqlda->sqlvar[col].sqldata);   /*  save
data and advance */
            listbufnxt += saved_length;
/*  advance past value     */
            break;

          case SQL_TYP_CSTR:
/*  null-terminated ... but we don't want terminator
*/
            *sav_lengthptr = --saved_length;
/*  save length excluding null    */
            memcpy((void *)listbufnxt, (void
*)col_dataptr, saved_length); /*  copy string without
terminating null */
            listbufnxt += saved_length;
/*  advance past string     */
            break;

            /* all of the following don't include a
terminating null in the data   */
          case SQL_TYP_DATE:
          case SQL_TYP_TIME:
          case SQL_TYP_STAMP:
          case SQL_TYP_CHAR:
/*  fixed length  ...    */
            memcpy((void *)listbufnxt, (void
*)col_dataptr, saved_length); /*  doesn't include a
terminating null */
            *sav_lengthptr = saved_length;
/*  save length */
            listbufnxt += saved_length;
/*  advance past string and null     */
            break;

          case SQL_TYP_VARCHAR:
/*   variable ...                 */
          case SQL_TYP_LONG:
/*   ... length precedes ...      */
            saved_length = ((struct sqlchar *)sqlda-
>sqlvar[col].sqldata)->length;
            memcpy((void *)listbufnxt, (void
*)((struct sqlchar *)sqlda->sqlvar[col].sqldata)-
>data, saved_length);
            *sav_lengthptr = saved_length;
/*  save length */
            listbufnxt += saved_length;
/*  advance past string and null     */
            break;

          case SQL_TYP_DECIMAL:
          case SQL_TYP_NDECIMAL:
            /*   we must save the scale||precision in
the buffer - place them in the length field before the
value */
            *sav_lengthptr = saved_length;
/*  copy the scale||precision */
            /*  now calculate the byte-length of the
value */
            saved_length = ((*(struct declen *)&sqlda-
>sqlvar[col].sqllen).m + 1)/2;
            memcpy((void *)listbufnxt, (void
*)col_dataptr, saved_length); /*  copy the field
*/
            listbufnxt += saved_length;
/*  advance past saved value     */
            break;

          default:
            fprintf(stderr,"save_sqlda: Unknown column
type (%d)\n",col_type); fflush(stderr);
            break;
        }
      }   /*  end not null   */
   }
   return;
}

#else  /*   NO_PARALLEL_FORMAT_FETCH  */
/************************************************
*********************/
/* 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;
}
#endif /*   NO_PARALLEL_FORMAT_FETCH  */

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

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

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

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



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


#if defined(SQLSNI)
  status = gettimeofday(&end_time);
#elif defined(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;
   LPRINTF(outstream,"********************  DUMP OF
SQLCA  ********************\n");
   LPRINTF(outstream,"SQLCAID   : %.8s\n", ca-
>sqlcaid);
   LPRINTF(outstream,"SQLCABC   : %d\n", ca->sqlcabc);
   LPRINTF(outstream,"SQLCODE   : %d\n", ca->sqlcode);
   LPRINTF(outstream,"SQLERRML  : %d\n", ca-
>sqlerrml);
   LPRINTF(outstream,"SQLERRMC  : %.*s\n", ca-
>sqlerrml, ca->sqlerrmc);
   LPRINTF(outstream,"SQLERRP   : %.8s\n", ca-
>sqlerrp);

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

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

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

```
         }

         LPRINTF(outstream, "SQLWARN  = %0.11s\n",
temp_sqlca.sqlwarn);
         LPRINTF(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
   char *charptr;

   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 ((charptr = getenv("TPCD_LOG")) &&
             (!strcmp(uppercase(charptr),"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) {
              if(verbose) {
                 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;
   char *charptr;

   /*  get vmstat and iostat output file path */
   if (   (charptr = getenv("VMSTATOUTPATH"))
       && (g_struct->vmstat_out_path =
malloc((strlen(charptr) + 1)))
      )
   {
     strcpy(g_struct->vmstat_out_path, charptr);
   }

   if (   (charptr = getenv("IOSTATOUTPATH"))
       && (g_struct->iostat_out_path =
malloc((strlen(charptr) + 1)))
      )
   {
     strcpy(g_struct->iostat_out_path, charptr);
   }

   /** 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(stderr, "XXThe input file could not
be opened.\n\n");
         fprintf(stderr,"Make sure that the filename
is correct.\n");
         fprintf(stderr,"filename = %s\n",g_struct-
>c_l_opt->infile);
         fflush(stderr);
         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 ((charptr = getenv("TPCD_LOG")) &&
       (!strcmp(uppercase(charptr),"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 (charptr = getenv("TPCD_RUN_DIR"))
      strcpy(g_struct->run_dir,charptr);
   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);
     fflush(stderr);
     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: stream %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    */

   LPRINTF(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;
      }

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

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

   LPRINTF(outstream,"\n");
   for (loopvar=0; loopvar < total_length; loopvar++)
      LPRINTF(outstream,"-");
   LPRINTF(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_stam
p.tv_usec);
#elif (defined (SQLOS2) || defined(SQLWINT) || defined
(SQLWIN) || defined(SQLDOS))
         sprintf(newtime+timeLength,".%0.3d",temp_stam
p.millitm);
#else
#error Unknown operating system
#endif
         /* TIME_ACC jen stop*/
      }
   }
   else
      if (form == T_STAMP_FORM_2)
         strftime(newtime,50,"%y%m%d%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_ti
me + 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
                                                number
s < 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   %s\n\nStream Ts %11 = %10.0f\n\nStream start
int representation %11 = %f\n\nStream stop int
representation  %11 = %f",
                adjusted_g_mean_intern
                ,(g_struct->run_encountered_error ?
"run encountered error" : "")
                ,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 )
{
#ifdef TEMPORARILY_NOT_SPAWNING
     int rc;
#endif

#ifdef TPCD_PLOAD_SCRIPTS
   char statement[3000];        /*  system command to
run perl script */
#endif
   char sourcedir[256];


   int split_updates = 2;       /* no. of ways update
records are split */
   int concurrent_inserts = 0;  /* 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];
   int                  sleep_counter;
#else
   int childpid[100];
   int                  su_semid; /* semaphore for
controlling split updates*/
   key_t                su_semkey; /* key to generate
semid */
#endif

   Timer_struct curtime = { 0 , 0 };
   double load_elapsed_time;
#ifdef RECORD_START_OF_LOAD
   Timer_struct endtime = { 0 , 0 };
   int st_load_fd;
   int st_load_nbytes;
   char st_load_buf[128];
#endif

   unsigned int inter_uf_child_msec;   /*  delay
between starting successive chunkers */
   char *charptr;

   /*   variables for controlling spawning concurrent
processes for the two loads  */
#ifndef TPCD_PLOAD_SCRIPTS
#ifndef SQLWINT
   pid_t forkrc, ord_pid, lin_pid;
#else
#ifndef TEMPORARILY_NOT_SPAWNING
   /*   ##Des##, please add whatever you need for
variables for controlling spawning here */
   HANDLE hThread_Parallel_Load;
#endif /*  TEMPORARILY_NOT_SPAWNING */
#endif
#endif /* TPCD_PLOAD_SCRIPTS */

   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 */
   /* two different tables are to be loaded - orders
into orders_staging and lineitems into lineitem
staging */
   /* for best performance, we will load these
concurrently   */
   if (env_tpcd_audit_dir != NULL)
      strcpy(sourcedir,env_tpcd_audit_dir);
   else
      strcpy(sourcedir,".");

   /* Load the orderkeys into the staging table */
   /*  choice of three methods - load API (preferred),
system command to run db2 clp, run perl script */
   get_start_time(&curtime);

#ifdef RECORD_START_OF_LOAD
   /*   record time of start-of-load in a hopefully-
uniquely-named file */
   /*  st_load_nbytes = sprintf(st_load_buf,
"/tmp/aardvark/runUF1:%d.%d", curtime.tv_sec,
curtime.tv_usec);  */
   strcpy(st_load_buf, "/tmp/");
   strcat(st_load_buf, getenv("USER"));
   st_load_nbytes = strlen(st_load_buf);
   st_load_nbytes +=
sprintf(st_load_buf+st_load_nbytes, "/runUF1:%d.%d",
curtime.tv_sec, curtime.tv_usec);
   if ((st_load_fd = open(st_load_buf,O_CREAT|
O_WRONLY, S_IRUSR+S_IWUSR+S_IRGRP+S_IWGRP)) >= 0)
   {
      st_load_buf[st_load_nbytes++] = '\n';   /*
append linefeed  */
      lseek(st_load_fd, 0, SEEK_END);         /*
just in case     */
      write(st_load_fd, st_load_buf, st_load_nbytes);
      close(st_load_fd);
   }
#endif /* RECORD_START_OF_LOAD */

#ifdef TPCD_PLOAD_SCRIPTS
#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);
   }
#else /* not  TPCD_PLOAD_SCRIPTS - do the work
directly in this function without calling additional
tpc-d kit scripts;
      **  note that because we want to load both
orders and lineitem concurrently (see above),
      **  we must fork/spawn processes and run them in
background (similar to what the tpc-d kit scripts do)
      */
   charptr=getenv("TPCD_FLATFILES");  /*  path for
input to load */
   /*   we must fork subprocesses to run each load,
otherwise they clash on trying to use same db2bp */
#ifndef SQLWINT
   /*  before forking new processes, flush outstream -
otherwise each new process duplicates what it
inherited in the buffer */
   fflush(outstream);
   if ((forkrc = fork()) == 0)
#else
#ifndef TEMPORARILY_NOT_SPAWNING
#ifdef TPCD_PLOAD_API
   /*   ##Des##, please add whatever you need for
spawning thread here */
   /* Kick off a new thread which will begin to load
the orders UF data.          */
   /* In parallel, the load of the lineitem UF data
will be started and afterwards */
   /* it will check and wait, if necessary for the
orders load to complete.      */
   /* The thread is terminated automatically after it
completes execution.        */
   ThreadParm = NULL;
   /* Following line spawns the thread and sets it
running at function Load_UF_Data_Orders. */
   Parallel_Load_Complete = FALSE;
   pupdatePair = updatePair;
   hThread_Parallel_Load = CreateThread(NULL, 32768,
Load_UF_Data_Orders, ThreadParm, 0,
&Thread_Parallel_Load_ID);
   if(hThread_Parallel_Load == NULL) {
     fprintf(stderr,"Failed to create parallel load
thread.\n");
     fflush(stderr);
     exit(1);
   }
   goto load_li_data;
#else /*  not TPCD_LOAD_API */
    error "running db2 clp under Windows not yet
supported - code needed"
#endif /* TPCD_LOAD_API */
#endif /*  TEMPORARILY_NOT_SPAWNING */
```

```
#endif
   {                                        /*   I am
child  */
#ifdef TPCD_PLOAD_API
     /*  use db2Load api - function source is in
tpcdUF.sqc  */
#ifdef TEMPORARILY_NOT_SPAWNING
     rc =
#else
     exit
#endif /*  TEMPORARILY_NOT_SPAWNING */
       (tpcd_load_staging((TPCDBATCH_INSERT+TPCDBATCH
_ORDERS),dbname,updatePair,charptr,verbose));
#ifdef TEMPORARILY_NOT_SPAWNING
     if(rc < 0)
     {
       fprintf(stderr,"Error %d Loading Orders UF
Data\n",rc);
       exit(0);
     }
#endif
#else /* use db2 CLP */
     char statemord[3000];
     sprintf (statemord, "print -- \"connect to
%s;\\\\nload from orders.tbl.u%d.new of del modified
by coldel| fastparse messages "
#ifdef TPCD_PLOAD_MESSAGES
         "%s/ord.msg.u%d.new"
#else
         "/dev/null"
#endif
         " replace into TPCDTEMP.ORDERS_NEW
nonrecoverable partitioned db config mode load_only
part_file_location %s;\\\\nconnect
reset;\\\\nterminate;\"|db2 -t +p"
               ,dbname,updatePair
#ifdef TPCD_PLOAD_MESSAGES
               ,charptr,updatePair
#endif
               ,charptr);
     if (verbose)    fprintf(stderr,"UF1 ord shell
statement %s\n",statemord);
     if (system(statemord))
     {
        fprintf (stderr, "db2 load failed for UF1 new
ord"
#ifdef TPCD_PLOAD_MESSAGES
                 ", examine %s/ord.msg.u%d.new for
cause"
#endif
                 ". Exiting.\n"
#ifdef TPCD_PLOAD_MESSAGES
                 ,charptr,updatePair
#endif
                 );
        exit (-1);
     }
     exit(0);
#endif /* use db2 CLP */
   }
load_li_data:
#ifndef SQLWINT
   ord_pid = forkrc;  /*  orders child process id */
   if (ord_pid > 0)   /*  successful so far ... */
#else
#ifndef TEMPORARILY_NOT_SPAWNING
   /*   ##Des##, please add whatever you need for
checking orders child successfully spawned here */
#endif /*  TEMPORARILY_NOT_SPAWNING */
#endif
   {
#ifndef SQLWINT
     if ((forkrc = fork()) == 0)
#else
#ifndef TEMPORARILY_NOT_SPAWNING
   /*   ##Des##, please add whatever you need for
spawning thread here */
#endif /*  TEMPORARILY_NOT_SPAWNING */
#endif  /*  WINNT */
     {                                      /*   I am
child  */
#ifdef TPCD_PLOAD_API
     /*  use db2Load api - function source is in
tpcdUF.sqc  */
#ifdef SQLWINT
#ifdef TEMPORARILY_NOT_SPAWNING
     rc =
#endif /*  TEMPORARILY_NOT_SPAWNING */
#else /*  not WINNT */
     exit
#endif /* not WINNT */
       (tpcd_load_staging((TPCDBATCH_INSERT+TPCDBATCH_
LINEITEM),dbname,updatePair,charptr,verbose));
#ifdef TEMPORARILY_NOT_SPAWNING
     if(rc < 0)
     {
       fprintf(stderr,"Error %d Loading Lineitem UF
Data\n",rc);
       fflush(stderr);
       exit(0);
     }
#endif
#else /* use db2 CLP */
       char statemlin[3000];
       sprintf (statemlin, "print -- \"connect to
%s;\\\\nload from lineitem.tbl.u%d.new of del modified
by coldel| fastparse messages "
#ifdef TPCD_PLOAD_MESSAGES
         "%s/lin.msg.u%d.new"
#else
         "/dev/null"
#endif
         " replace into TPCDTEMP.LINEITEM_NEW
nonrecoverable partitioned db config mode load_only
part_file_location %s;\\\\nconnect
reset;\\\\nterminate;\"|db2 -t +p"
               ,dbname,updatePair
#ifdef TPCD_PLOAD_MESSAGES
               ,charptr,updatePair
#endif
               ,charptr);
       if (verbose)    fprintf(stderr,"UF1 lin shell
statement %s\n",statemlin);

       if (system(statemlin))
       {
          fprintf (stderr, "db2 load failed for UF1
new lin"
#ifdef TPCD_PLOAD_MESSAGES
                   ", examine %s/lin.msg.u%d.new for
cause"
#endif
                   ". Exiting.\n"
#ifdef TPCD_PLOAD_MESSAGES
                   ,charptr,updatePair
#endif
                   );
          exit (-1);
       }
       exit(0);
#endif /* use db2 CLP */
     }
#ifndef SQLWINT
     lin_pid = forkrc;  /*  lineitem child process id
*/
#else
#ifndef TEMPORARILY_NOT_SPAWNING
   /*   ##Des##, please add whatever you need for
tracking lineitem thread here */
   /* Check if the UF orders data has loaded
successfully */
   sleep_counter = 0;
   if (verbose) {
      fprintf(stderr,"Waiting for Orders to load.\n");
      fflush(stderr);
   }
   while( Parallel_Load_Complete != TRUE) {
     if(sleep_counter != 6000) {     /* Check every
100 millisecs for 5 mins to complete    */
        sleep(100);      /* otherwise abort.
*/
        sleep_counter++;
     }
     else {
        fprintf(stderr,"Load of UF Orders din not
complete in time allotted.\n");
        fflush(stderr);
        exit(0);
     }
   }
   if(verbose) {
      fprintf(stderr,"Waiting for Orders to load -
```

```
Done.\n");
      fflush(stderr);
   }
   if(iParallel_Load_Result) {
     fprintf(stderr,"Load of UF Orders Failed.\n");
     fflush(stderr);
     exit(0);
   }

#endif /*  TEMPORARILY_NOT_SPAWNING */
#endif
   }
#ifndef SQLWINT
   if (forkrc < 0)
   {
     fprintf(stderr,"UF1 unable to fork child\n");
exit(-1);
   }
#endif
#ifdef TEMPORARILY_NOT_SPAWNING
    goto bypass_thread_checks;
#else
   /*   ##Des##, please add whatever you need for
checking lineitem child successfully spawned here */
#endif /*  TEMPORARILY_NOT_SPAWNING */
#ifndef SQLWINT
   if ((forkrc = waitpid(ord_pid, &i, 0)) != ord_pid)
     fprintf(stderr,"UF1 problem waiting for orders
child %d\n",ord_pid);
   else if (i = WEXITSTATUS(i))
     fprintf(stderr,"UF1 orders child %d exited with
non-zero status %d\n",ord_pid,i);
   else if ((forkrc = waitpid(lin_pid, &i, 0)) !=
lin_pid)
     fprintf(stderr,"UF1 problem waiting for lineitem
child %d",lin_pid);
   else if (i = WEXITSTATUS(i))
     fprintf(stderr,"UF1 lineitem child %d exited with
non-zero status %d\n",lin_pid,i);
   else
#else
#ifndef TEMPORARILY_NOT_SPAWNING
   /*   ##Des##, please add whatever you need for
waiting for child threads to complete here */
#endif /*  TEMPORARILY_NOT_SPAWNING */
#endif
   {    /* forked children successfully  */
   if (verbose)
   {
     fprintf(stderr,"UF1: staging_orders child %d
completed\n     staging_lineitem child %d completed\n"
#ifndef SQLWINT
                  ,ord_pid,lin_pid
#else
#ifdef TEMPORARILY_NOT_SPAWNING
                  ,0,0          /*  temporary to let
this compile */
#else
   /*   ##Des##, please add whatever you need for
variables for reporting spawning here */
#endif /*  TEMPORARILY_NOT_SPAWNING */
#endif
           );
   }
#endif /* not  TPCD_PLOAD_SCRIPTS  - fork processes
directly */

#ifdef SQLWINT
#ifdef TEMPORARILY_NOT_SPAWNING
  bypass_thread_checks: ;
#endif /*  TEMPORARILY_NOT_SPAWNING */
#endif

   load_elapsed_time = get_elapsed_time(&curtime);
#ifdef RECORD_START_OF_LOAD
   /*   record time of end-of-load in order to compute
load elapsed time */
   get_start_time(&endtime);
   endtime.tv_sec -= curtime.tv_sec;
   endtime.tv_usec -= curtime.tv_usec;
   if (endtime.tv_usec < 0)
   {
     endtime.tv_sec--;
     endtime.tv_usec += 1000000;
   }
#endif

   fprintf (outstream, "load staging for UF1 elapsed
time %15.1f secs"
#ifdef RECORD_START_OF_LOAD
    " %d.%09d"
#endif
     "\n",load_elapsed_time
#ifdef RECORD_START_OF_LOAD
      ,endtime.tv_sec,endtime.tv_usec
#endif
          );

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

   /*  skip the inserts into base tables if
TPCD_CONCURRENT_INSERTS is zero */
   if (concurrent_inserts > 0)
   {
#ifndef SQLWINT
   /*    we will use the tpcd.setup file to generate
the semaphore key */
      /* this is assuming that you will be running
this from 0th node */
      sprintf(sourcefile,
"%s%ctools%ctpcd.setup",env_tpcd_audit_dir,
PATH_DELIM,PATH_DELIM);
   /*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
      inter_uf_child_msec = ((unsigned
int)(inter_uf_child_delay*1000000.0));  /* microsec
for unix usleep() */
      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]);
      }
      if (inter_uf_child_delay > 0.0)
```

```
        usleep (inter_uf_child_msec);
#else  /* SQLWINT */
      inter_uf_child_msec = ((unsigned
int)(inter_uf_child_delay*1000.0));     /* millsec for
WinNT Sleep() */
      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);
      if (inter_uf_child_delay > 0.0)
        Sleep (inter_uf_child_msec);
#endif /* SQLWINT */
   }

   /* 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,"\nYY The output file could not
be opened.  ");
      fprintf(stderr,"Make sure that the filename is
correct.\n");
      fprintf(stderr,"filename =
%s\n",outstreamfilename);
      fflush(stderr);
      exit(-1);
   }

   fprintf( outstream,"UF1 for update pair %d
complete\n",updatePair);
   }  /*  end if concurrent deletes  */
#ifndef TPCD_PLOAD_SCRIPTS
   }  /*  end if forked children successfully  */
#endif
}


/* 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 )
{
#ifndef TPCD_PLOAD_API
   char statement[3000];        /*  system command to
run perl script or db2 clp  */
#endif
   char sourcedir[256];

   int split_deletes = 1;         /* no. of ways
update records are split   @dxxxxxhar */
   int concurrent_deletes  = 0;   /* 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

   Timer_struct curtime = { 0 , 0 };
   double load_elapsed_time;
#ifdef RECORD_START_OF_LOAD
   Timer_struct endtime = { 0 , 0 };
   int st_load_fd;
   int st_load_nbytes;
   char st_load_buf[128];
#endif

   unsigned int inter_uf_child_msec;   /*  delay
between starting successive chunkers */
   char *charptr;

    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 (charptr = getenv ("TPCD_SPLIT_DELETES"))
      split_deletes = atoi (charptr);
   /* How many deletes should run concurrently */
   if (charptr = getenv ("TPCD_CONCURRENT_DELETES"))
      concurrent_deletes = atoi (charptr);
   /* 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 */
   /*  choice of three methods - load API (preferred),
system command to run db2 clp, run perl script */
   get_start_time(&curtime);

#ifdef RECORD_START_OF_LOAD
   /*   record time of start-of-load in a hopefully-
uniquely-named file */
   /*  st_load_nbytes = sprintf(st_load_buf,
"/tmp/aardvark/runUF2:%d.%d", curtime.tv_sec,
curtime.tv_usec);  */
   strcpy(st_load_buf, "/tmp/");
   strcat(st_load_buf, getenv("USER"));
   st_load_nbytes = strlen(st_load_buf);
   st_load_nbytes +=
sprintf(st_load_buf+st_load_nbytes, "/runUF2:%d.%d",
curtime.tv_sec, curtime.tv_usec);
   if ((st_load_fd = open(st_load_buf,O_CREAT|
O_WRONLY, S_IRUSR+S_IWUSR+S_IRGRP+S_IWGRP)) >= 0)
   {
      st_load_buf[st_load_nbytes++] = '\n';   /*
append linefeed  */
      lseek(st_load_fd, 0, SEEK_END);         /*
just in case     */
      write(st_load_fd, st_load_buf, st_load_nbytes);
      close(st_load_fd);
   }
#endif /* RECORD_START_OF_LOAD */

#ifdef TPCD_PLOAD_SCRIPTS
#ifdef SQLWINT
   sprintf (statement, "perl %s\\tools\\ploaduf2
%d\n", sourcedir, updatePair);
#else
   sprintf (statement, "perl %s/tools/ploaduf2 %d 2",
sourcedir, updatePair);
#endif /* not SQLWINT */
#else /*  run db2 process directly */
   charptr=getenv("TPCD_FLATFILES");
#ifdef TPCD_PLOAD_API
     /*  use db2Load api - function source is in
tpcdUF.sqc  */
     i =
tpcd_load_staging((TPCDBATCH_DELETE+TPCDBATCH_ORDERS),
dbname,updatePair,charptr,verbose);
#else /* use db2 CLP */
   sprintf (statement, "print -- \"connect to
%s;\\\\nload from delete.%d.new of del modified by
coldel| fastparse messages "
#ifdef TPCD_PLOAD_MESSAGES
         "%s/rf2.msg.u%d.del"
#else
         "/dev/null"
#endif
         " replace into TPCDTEMP.ORDERS_DEL
nonrecoverable partitioned db config mode load_only
part_file_location %s;\\\\nconnect
reset;\\\\nterminate;\"|db2 -t +p"
              ,dbname,updatePair
#ifdef TPCD_PLOAD_MESSAGES
              ,charptr,updatePair
#endif
              ,charptr);
#endif /* use db2 CLP */
#endif /*  run db2 process directly */

#ifdef TPCD_PLOAD_API
   if (i)
#else /* use db2 CLP */
   if (verbose)    fprintf(stderr,"UF2 shell statement
%s\n",statement);
   if (system(statement))
#endif
   {
#ifdef TPCD_PLOAD_SCRIPTS
      fprintf (stderr, "ploaduf2 failed for UF2,
examine UF2.log for cause. Exiting.\n");
#else /*  run db2 process directly */
      fprintf (stderr, "db2 load failed for UF2 del"
#ifdef TPCD_PLOAD_MESSAGES
               ", examine %s/rf2.msg.u%d.del for
cause"
#endif
               ". Exiting.\n"
#ifdef TPCD_PLOAD_MESSAGES
               ,charptr,updatePair
#endif
               );
#endif /*  run db2 process directly */
      exit (-1);
   }
   load_elapsed_time = get_elapsed_time(&curtime);
#ifdef RECORD_START_OF_LOAD
   /*   record time of end-of-load in order to compute
load elapsed time */
   get_start_time(&endtime);
   endtime.tv_sec -= curtime.tv_sec;
   endtime.tv_usec -= curtime.tv_usec;
   if (endtime.tv_usec < 0)
   {
     endtime.tv_sec--;
     endtime.tv_usec += 1000000;
   }
#endif
   fprintf (outstream, "load staging for UF2 elapsed
time %15.1f secs"
#ifdef RECORD_START_OF_LOAD
        " %d.%09d"
#endif
            "\n",load_elapsed_time
#ifdef RECORD_START_OF_LOAD
      ,endtime.tv_sec,endtime.tv_usec
#endif
          );

   /* Next we need to get ready to launch a bunch of
concurrent processes */
   /* however, skip it if TPCD_CONCURRENT_DELETES is
set to 0 */
   if (concurrent_deletes > 0)
   {

   fclose(outstream); /* to prevent multiple header
caused by forking
                         wlc 081397 */
#ifndef SQLWINT
   /*    we will use the tpcd.setup file to generate
the semaphore key    begin SEMA */
   sprintf(sourcefile,
"%s%ctools%ctpcd.setup",env_tpcd_audit_dir,
PATH_DELIM, PATH_DELIM);
   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
     inter_uf_child_msec = ((unsigned
int)(inter_uf_child_delay*1000000.0));  /* microsec
for unix usleep() */
      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]);
      }
      if (inter_uf_child_delay > 0.0)
        usleep (inter_uf_child_msec);
#else
      {
         /*  SECURITY_ATTRIBUTES sec_process;
             SECURITY_ATTRIBUTES sec_thread; */
         /* NEED TO FIX THIS UP - KBS 98/10/20 */

        inter_uf_child_msec = ((unsigned
int)(inter_uf_child_delay*1000.0));     /* millsec for
WinNT Sleep() */
         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 */
         if(verbose) {
            fprintf(stderr, "commandline= %s\n",
commandline);
         }
         system (commandline);
         if (inter_uf_child_delay > 0.0)
           Sleep (inter_uf_child_msec);
      }
#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,"\nZZ The output file could not
be opened.  ");
      fprintf(stderr,"Make sure that the filename is
correct.\n");
      fprintf(stderr,"filename =
%s\n",outstreamfilename);
      fflush(stderr);
      exit(-1);
   }

   fprintf( outstream,"UF2 for update pair %d
complete\n",updatePair);
   }  /*  end if concurrent deletes  */
}


/* 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 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,"\nWW The input file could not
be opened.");
         fprintf(stderr,"Make sure that the filename
is correct.\n");
```

```
         fprintf(stderr,"filename = %s\n",g_struct-
>c_l_opt->infile);
         fflush(stderr);
         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");
         fflush(stderr);
         exit(-1);
      }

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

   g_struct->qnum = qnum;

   switch (g_struct->c_l_opt->intStreamNum)
   {
    case -1: /* qualifying */
      sprintf(outstreamfilename,
"%s%sqryqual%02d.%s",run_dir,delim,qnum,time_stamp);
      break;
    case 0: /* power tests */
       if (qnum < 0) /* update functions */
         sprintf(outstreamfilename,
"%s%smps00uf%d.%02d.%s",run_dir,delim,abs(qnum), \
                currentUpdatePair,time_stamp);
     else
         sprintf(outstreamfilename,
"%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(outstreamfilename,
"%s%smts%02duf%d.%02d.%s",run_dir,delim, \
                currentUpdatePair - updatePairStart +
1,abs(qnum), currentUpdatePair,time_stamp);
      else
         sprintf(outstreamfilename,
"%s%smts%dqry%02d.%s",run_dir,delim, \
                g_struct->c_l_opt-
>intStreamNum,qnum,time_stamp);
      break;

   }
#ifdef DES_DEBUG
   printf("OutfileNamexx = %s.\n", outstreamfilename);
   fflush(stdout);
#endif

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

      /*    Create an output file only if:
       *     .     there are input statements left to
process
       *       and no slave_formatter willl write its
output report
       *  or .     we have detected eof on input
stream
       *       and want to print out the summary table
file
       */
#ifdef DEBUG_OUTPUT_FILE
    fprintf(stderr,"pid %d instream= %X , feof=%X
eo_infile= %d intStreamNum= %d qnum= %d update= %d
about to create output file %s\n"
            ,getpid() ,instream ,feof(instream),
g_struct->c_flags->eo_infile ,g_struct->c_l_opt-
>intStreamNum ,qnum ,g_struct->c_l_opt->update
,outstreamfilename);  fflush(stderr);
#endif
   if (    (    (!feof(instream))
#ifndef NO_PARALLEL_FORMAT_FETCH      /*   parallel
format-fetch only if requested */
             && (    (    (g_struct->c_l_opt-
>intStreamNum == 0)    /*  power run ...         */
                         && (qnum < 0)
/*  ... update statement  */
                       )
                   || (    (g_struct->c_l_opt-
>intStreamNum  > 0)    /*  throughput stream ... */
                         && ((g_struct->c_l_opt->update
== 2) || (g_struct->c_l_opt->update == 5)) /* ... with
updates */
                       )
                  )
#endif /* ifndef NO_PARALLEL_FORMAT_FETCH  parallel
format-fetch only if requested */
               )
        || (g_struct->c_flags->eo_infile)
      )
      if( (outstream = fopen(outstreamfilename,
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",outstreamfilename);
         fflush(stderr);
         exit(-1);
      }
```

```
      else if (verbose) fprintf(stderr,"main proc pid
%d opened %s\n",getpid() ,outstreamfilename);

   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 )
      {
#ifdef DEBUG_SEMCONTROL
       fprintf(stderr,"PreSQLprocess INSERT pid %d
update= %d currentUpdatePair= %d updatePairStart= %d
intStreamNum= %d\n"
             ,getpid() ,g_struct->c_l_opt->update
,currentUpdatePair ,updatePairStart ,g_struct-
>c_l_opt->intStreamNum);
       fflush(stderr);
#endif
        /* 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_tim
e ));  /* TIME_ACC jen*/
        /* write the start timestamp to the file...if
this is not a qualification */
        /* run, then write the seed used as well */
        LPRINTF(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)
          {
            LPRINTF(outstream,"Using default qgen seed
file");
          }
          else
            LPRINTF(outstream,"Seed used =
%ld",g_struct->lSeed);
          LPRINTF(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);
#ifdef DEBUG_SEMCONTROL
      if ( semcontrol == 1 )
      {
        fprintf(stderr,"PreSQLprocess INSERT pid %d
update= %d currentUpdatePair= %d updatePairStart= %d
intStreamNum= %d pausing\n"
              ,getpid() ,g_struct->c_l_opt->update
,currentUpdatePair ,updatePairStart ,g_struct-
>c_l_opt->intStreamNum);
        fflush(stderr);
        pause();
      }
#endif
      break;
    case TPCDBATCH_DELETE:
#ifdef DEBUG_SEMCONTROL
      if ( semcontrol == 1 )
      {
        fprintf(stderr,"PreSQLprocess DELETE pid %d
pre-wait update= %d currentUpdatePair= %d
updatePairStart= %d intStreamNum= %d\n"
              ,getpid() ,g_struct->c_l_opt->update
,currentUpdatePair ,updatePairStart ,g_struct-
>c_l_opt->intStreamNum);
        fflush(stderr);
      }
#endif
      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);
      }
#ifdef DEBUG_SEMCONTROL
      if ( semcontrol == 1 )
      {
        fprintf(stderr,"PreSQLprocess DELETE pid %d
post-wait update= %d currentUpdatePair= %d
updatePairStart= %d intStreamNum= %d\n"
              ,getpid() ,g_struct->c_l_opt->update
,currentUpdatePair ,updatePairStart ,g_struct-
>c_l_opt->intStreamNum);
        fflush(stderr);
      }
#endif
      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_tim
e ));  /* TIME_ACC jen*/
        /* write the start timestamp to the file...if
this is not a qualification */
        /* run, then write the seed used as well */
        LPRINTF(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)
          {
            LPRINTF(outstream,"Using default qgen seed
file");
          }
          else
            LPRINTF(outstream,"Seed used =
%ld",g_struct->lSeed);
          LPRINTF(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 */
   long   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
                                                   column
s in returned set  */
   struct stmt_info *s_info_ptr;

   FILE *vmstat_file;
   FILE *iostat_file;

   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");
            g_struct->run_encountered_error=1;  /* for
reporting metrics        */
            rc = -1;
         }
         else  /* print out the column headings for
the answer set */
         {
            print_headings(sqlda,col_lengths);
/* @d22817 tjg */

            /*   insert query start point into vmstat
output file          jel*/
            if (   (g_struct->vmstat_out_path)
                && (vmstat_file = fopen(g_struct-
>vmstat_out_path,APPENDMODE))
               )
            {
               fprintf(vmstat_file,"\nQuery %d of
stream %d starting at : %18.18s \n"
                     ,g_struct->qnum ,g_struct->c_l_opt-
>intStreamNum ,s_info_ptr->start_stamp );
               fclose(vmstat_file);
            }
```

```
            if (   (g_struct->iostat_out_path)
                && (iostat_file = fopen(g_struct-
>iostat_out_path,APPENDMODE))
               )
            {
               fprintf(iostat_file,"\nQuery %d of
stream %d starting at : %18.18s \n"
                     ,g_struct->qnum ,g_struct->c_l_opt-
>intStreamNum ,s_info_ptr->start_stamp );
               fclose(iostat_file);
            }

            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");
               g_struct->run_encountered_error=1;  /*
for reporting metrics        */
               SleepSome(10);
            }
            else
            {
               /*************************************
*********************************/
               /* Fetch appropriate number of rows and
determine whether or not to    */
               /* send them to file.
*/
               /*************************************
*********************************/

               rows_fetch = 0;

#ifndef NO_PARALLEL_FORMAT_FETCH      /*   parallel
format-fetch only if requested */
               if (listbufnxt)
               {
                  listbufcurhdp=listbufnxt;    /*
remember end of header */
                  /*   save sqld and the tpcdbatch
column formatting lengths */
                  *((short *)listbufnxt) = sqlda->sqld;
listbufnxt += 2;     /*  save sqld and advance */
                  memcpy((void *)listbufnxt,(void
*)col_lengths, (sqlda->sqld *
sizeof(col_lengths[0])));
                  listbufnxt += (sqlda->sqld *
sizeof(col_lengths[0]));
               }
#endif  /*  NO_PARALLEL_FORMAT_FETCH      parallel
format-fetch only if requested */

               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) )
#ifndef NO_PARALLEL_FORMAT_FETCH      /*   parallel
format-fetch only if requested */
                        save_sqlda(sqlda);
#else  /*   NO_PARALLEL_FORMAT_FETCH  */
                        echo_sqlda(sqlda,col_lengths);
#endif /*   NO_PARALLEL_FORMAT_FETCH  */
                  }
                  else if (sqlcode < 0)
                  {
                     max_wait++;
                     fprintf (stderr, "\nAn error has
been detected on fetch...Retrying...\n");
                     g_struct-
>run_encountered_error=1;  /* for reporting metrics
*/
                     SleepSome(10);
                  }
               }  while ( (sqlcode == 0) && \
                          ( (s_info_ptr-
>max_rows_fetch == -1) || \
                            (rows_fetch < s_info_ptr-
>max_rows_fetch) ) );
               if (   ((sqlcode < 0) ||
( (sqlca.sqlcode > 0) && (rows_fetch == 0) ))
                    && (g_struct->c_flags-
>select_status != TPCDBATCH_OPT_DRIVER) /*  ignore
errors from opt driver commands */
                  )
                  g_struct->run_encountered_error=1;
/* for reporting metrics        */
            } /* end of successful open */
         } /* end of successful prepare */
#ifndef NO_PARALLEL_FORMAT_FETCH      /*   parallel
format-fetch only if requested */
         listbufcurtrp=listbufnxt;    /*  remember end
of rows */
#ifdef TPCD_PROGRESS_FILE
         if (listbufctlp)
         {                            /*  remember
sqlcode and rowcount */
           s_info_ptr->rows_fetch = rows_fetch;
           s_info_ptr->sqlcode = sqlcode;
         }
#endif
#endif  /*  NO_PARALLEL_FORMAT_FETCH      parallel
format-fetch only if requested */
      } /** 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");
            g_struct->run_encountered_error=1;  /* for
reporting metrics        */
            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
            LPRINTF(outstream, "\n\nNumber of rows
retrieved is: %6d",
                    rows_fetch);
```

```
         else
            LPRINTF(outstream, "\n\nNumber of rows
retrieved is: %6d",
                    s_info_ptr->max_rows_fetch);
#else
         LPRINTF(outstream, "\n\nNumber of rows
retrieved is: %6d",
                 rows_fetch);
         else
            LPRINTF(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 */
   int rc = TRUE;                     /*  optimistic
assumption that we want to continue */
#ifndef NO_PARALLEL_FORMAT_FETCH      /*  parallel
format-fetch only if requested */
    int temp_rc;                      /*  rc from
parent_rendezvous_point */
#endif
#ifdef TPCD_DB2SPEC
   char spec_buff[64];
#endif

#if DEBUG
   LPRINTF(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 */
     }
   }
#ifdef TPCD_DB2SPEC
   sprintf(spec_buff,"print -- \"GFDB\\\\nBPSDBCB
PFWTQ%d TF\\\\nQ\\\\n\"|db2spec",g_struct->qnum);
   system(spec_buff);
#endif
   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 stop timestamp to the file */
   LPRINTF(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  */
   LPRINTF(outstream,"Query Time = %15.1f secs\n",
s_info_ptr->elapse_time);

#ifndef NO_PARALLEL_FORMAT_FETCH      /*   parallel
format-fetch only if requested */
   if (g_struct->c_l_opt->update < 2) /*   no parallel
format-fetch if running updates */
   {
     if (slave_formatter_pid)       /*  if our
slave_formatter is waiting to write to file */
      {
#ifndef CHECK_EVERY_BUF_STORE
       /*  check that the shared-mem buffer has not
been overwritten - die if it has */
       if ((listbufnxt - listbufcur) >
RND_LISTBUFSIZE)
       {
         fprintf(stderr,"OOOPPPSSS   parent wrote %ld
bytes into sharedmem buffer of size %ld, increase by
-DLISTBUFSIZE=0x%pL\n"
                 ,(long)(listbufnxt - listbufcur),
(long)RND_LISTBUFSIZE ,((char *)((long)( 1.5 *
(double)(listbufnxt - listbufcur) ))));
fflush(stderr);
        listbufcur = 0;         /*  tell slave to
terminate  */
        rc = FALSE;             /*  tell myself and
slave to terminate */
       }
#endif
       /*************************************************
*****************/
       /*    rendezvous
*/
       /*    ensure slave has finished formatting
previous query      */
       /*    before proceeding with next one.
*/
       /*************************************************
*****************/
       if (temp_rc =
parent_rendezvous_point(g_struct,rc)) /* rendezvous
and tell slave to proceed (or terminate) */
       {
         fprintf(stderr,"parent rendezvous failed rc
%d\n",temp_rc);
        rc = FALSE;
       }
     }
     else rc = FALSE;   /*   probably best to shut
down if supposed to be doing parallel format/fetch and
no slave_formatter_pid */
   }
#endif  /*  NO_PARALLEL_FORMAT_FETCH      parallel
format-fetch only if requested */

   /** 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)
#ifndef NO_PARALLEL_FORMAT_FETCH      /*   parallel
format-fetch only if requested */
        && (!slave_formatter_pid)       /* no
slave_formatter writing to file           */
#endif /* ifndef NO_PARALLEL_FORMAT_FETCH  parallel
format-fetch only if requested */
      )
      fclose(outstream);

   return rc;
}

/*************************************************
************************/
/* 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)
      {
#ifdef DEBUG_SEMCONTROL
       fprintf(stderr,"pid %d intStreamNum= %d update=
%d about to release query power sem\n"
            ,getpid() ,g_struct->c_l_opt->intStreamNum
,g_struct->c_l_opt->update);  fflush(stderr);
#endif
       release_semaphore(g_struct, QUERY_POWER_SEM);
/* power stream */
      }
      else
      {
#ifdef DEBUG_SEMCONTROL
       fprintf(stderr,"pid %d intStreamNum= %d update=
%d about to release throughput sem\n"
            ,getpid() ,g_struct->c_l_opt->intStreamNum
,g_struct->c_l_opt->update);  fflush(stderr);
#endif
       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_stru
ct->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
,
                                           "Throu
ghput 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
,
                                                "Throu
ghput 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
,
                                                "Throu
ghput 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 pid %d waiting for
update stream (UF1) to signal semaphore based on
```

```
%s\n", getpid() , semfile);
  else
     fprintf(stderr,"updatestream pid %d (UF2) waiting
on querystream semaphore to signal semaphore based on
%s\n", getpid() , semfile);
  fflush(stderr);

#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 pid %d finished
waiting on updatestream semaphore\n", getpid());
    else
     fprintf(stderr,"updatestream pid %d finished
waiting on querystream semaphore\n", getpid());
  fflush(stderr);
}

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

            Sleep(1000);
        }
        return hSem;
}

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

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

        semkey = ftok(g_struct->update_num_file,'J');
```

```
        while ((semid = semget(semkey,num,0)) < 0)
        {
                if (errno == ENOENT)
                {
                        sleep(2);
                        fprintf(stderr,"cleanUp:
looping for access to semaphore stream %d  ",
                                g_struct->c_l_opt-
>intStreamNum);
                        fprintf(stderr,"semkey=%ld
semid = %d file=%s\n",semkey,semid,
                                g_struct-
>update_num_file);
                }
                else
                {
                        fprintf(stderr,"query stream
%d semget failed errno = %d\n",
                                g_struct->c_l_opt-
>intStreamNum,errno);
                        exit(1);
                }
        }
        return semid;
}
#endif
```

tpcduf.sqc

```
/*
########################################################
####################### */
/* ## Licensed Materials - Property of IBM
*/
/* ##
*/
/* ## Governed under the terms of the International
*/
/* ## License Agreement for Non-Warranted Sample Code.
*/
/* ##
*/
/* ## (C) COPYRIGHT International Business Machines
Corp. 1990 - 2005            */
/* ## All Rights Reserved.
*/
/* ##
*/
/* ## US Government Users Restricted Rights - Use,
duplication or                 */
/* ## disclosure restricted by GSA ADP Schedule
Contract with IBM Corp.          */
/*
########################################################
####################### */
/*******************************************************
**********************
*
*  TPCDUF.SQC
*
*  Revision History:
*
*  05 dec 98 aph Created tpcdUF.sqc containing
runUF1_fn() and runUF2_fn()
*                so that it can be bound separately
with a different isolation level.
*  15 may 99 bbe Added cast (short) for type
conversion between a long and a short.
*  16 jun 99 jen Added in proper connect reset code
for UF functions (mistakenly
*                removed
*  17 jun 99 jen SEMA Changes semaphore file for
update functions to look for tpcd.setup
*                not for the orders.***  update data
file  (AIX  only )
*  21 jul 99 bbe Commented out conditions in SQL
statments that searched on fields
*                other than app_id.
*
*******************************************************
**********************/

/******    define these if you want extra debugging
....
#define UF1DEBUG
#define UF2DEBUG
 ******/
#if (defined(UF1DEBUG) || defined(UF2DEBUG))
#define ALLDEBUG
#endif

#if (defined(SQLSUN))
#define exit(rc) _exit(rc)
#else
#define exit(rc) exit(rc)
#endif /* SQLPTX & SQLSUN*/

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

#include "sqlca.h"
#include <sqlenv.h>
#include <sqlutil.h>
/*  note concerning compilation of the db2Load api
with different versions of db2 :
**       the db2Load api takes as first param
"versionNumber", a db2 version number.
**       this versionNumber describes the remaining
parameters,
**       i.e. it tells the db2 server which set of
parameters are supposed to be set/not set
**       It is supposed to be safe to set it to any
version equal to or earlier than
**       the version in use during
compilation/execution provided that only parameters
**       defined as of the versionNumber version are
set.
**       So in theory you can set this to 815 since I
have run it on 815 and it works.
**       However, it may be safer to set it to the
current release.
**       This is difficult to do since you may not
know what value (symbol or constant) to specify.
**       To make it easier, I have provided a way of
doing so:
**         . in the makefile, specify
-DSQLZ_CURRENT_RELEASE
**         . copy the file sqlzrelease.h from the
current build's .../engn/include
**                                        into the same
directory as this file (your tpcdUF.sqc)
**       This will result in the following code using
the current versionNumber defined in sqlzrelease.h
**       As an alternative, you can specify
**       -DSQLZ_CURRENT_RELEASE=<literal>
**       and the program will then use that release
*/
#if ( defined(SQLZ_CURRENT_RELEASE) &&
(SQLZ_CURRENT_RELEASE == 1) )
#undef SQLZ_CURRENT_RELEASE    /* undefine in order to
use what's in the following ... */
#include "sqlzrelease.h"       /*  defines
SQLZ_CURRENT_RELEASE  */
#endif

#include <db2ApiDf.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_struc
t *)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_SHIPPRIORITY,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_QUANTITY
,
               L_EXTENDEDPRICE,L_DISCOUNT,L_TAX,
               L_RETURNFLAG,L_LINESTATUS,L_SHIPDATE,L_
COMMITDATE,L_RECEIPTDATE,
               L_SHIPINSTRUCT,L_SHIPMODE,L_COMMENT
               FROM TPCDTEMP.LINEITEM_NEW WHERE APP_ID
= :UF_chunk;
               /*(AND L_ORDERKEY IN
                 (SELECT O_ORDERKEY FROM TPCD.ORDERS
                    WHERE 12*(YEAR(O_ORDERDATE)-
1992)+MONTH(O_ORDERDATE)-01 = :month);*/

            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"));
#ifdef UF1DEBUG
      fprintf(stderr,"UF1 semfile =
%s\n",UF1_semfile);
#endif
   while ((su_hSem =
OpenSemaphore(SEMAPHORE_ALL_ACCESS |
                                    SEMAPHORE_MODIFY_ST
ATE |
                                    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,thisConcurre
ntDelete);
      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_struc
t *)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_STA
MP_1LEN,T_STAMP_1LEN,  /* TIME_ACC jen*/
                        get_time_stamp(T_STAMP_FORM_1,(
Timer_struct *)NULL));  /* TIME_ACC jen*/
               fprintf( myoutstream,
                        "    deletes for update pair
%d chunk %d complete at %*.*s\n\n",
                        updatePair, UF_chunk,
                        T_STAMP_1LEN,T_STAMP_1LEN,  /*
TIME_ACC jen*/
                        get_time_stamp(T_STAMP_FORM_1,
```

```
                                        (Timer_struct
*)NULL));  /* TIME_ACC jen*/
#endif
             }
         } /* process orders  deletes */
      } /* while trying to delete one chunk loop */
   } /* while there are more chunks */

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

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

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


UF2_exit:
   fclose (myoutstream);

   /* exiting, increment the semaphore */
#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_ST
ATE |
                                     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 */
}

/* PrintLoadSummary prints the load summary plus the
sqlcode message that was returned by LOAD */
void PrintLoadSummary(db2LoadOut     *pLoadInfoOut,
                      db2PartLoadOut
*pPartLoadInfoOut,
                      struct sqlca   *pSqlca)
{
  int i;
  char *loadAgentName[] = {"LOAD_AGENT",
                           "PARTITIONING_AGENT",
                           "PRE_PARTITIONING_AGENT",
                           "FILE_TRANSFER_AGENT",
                           "LOAD_TO_FILE_AGENT"};
  int numAgentInfoEntries;

  /* Determine the number of agent info entries in the
list.  If we   */
  /* didn't allocate enough memory,
oNumAgentInfoEntries could be     */
  /* greater than iMaxAgentInfoEntries, but in this
case we should    */
  /* only display the first iMaxAgentInfoEntries
elements of the list */
  if (pPartLoadInfoOut->oNumAgentInfoEntries <
      pPartLoadInfoOut->iMaxAgentInfoEntries)
  {
    numAgentInfoEntries = pPartLoadInfoOut-
>oNumAgentInfoEntries;
  }
  else
  {
    numAgentInfoEntries = pPartLoadInfoOut-
>iMaxAgentInfoEntries;
  }

  fprintf(stderr,"\nRESULTS OF LOAD OPERATION:\n\n");

  fprintf(stderr,"  LOAD AGENT TYPE           NODE
SQLCODE TABLE STATE     \n");
  fprintf(stderr,"
-------------------------------------------------------
-\n");

  /* First dump some summary information about the
partitioned db load */
  for (i = 0; i < numAgentInfoEntries; i++)
  {
    fprintf(stderr,"  %-25s  %3.3d  %+6d ",
            loadAgentName[pPartLoadInfoOut-
>poAgentInfoList[i].oAgentType],
            pPartLoadInfoOut-
```

```
>poAgentInfoList[i].oNodeNum,
           pPartLoadInfoOut-
>poAgentInfoList[i].oSqlcode);

    /* Display the table state on loading partitions
*/
    if (pPartLoadInfoOut-
>poAgentInfoList[i].oAgentType ==
        DB2LOAD_LOAD_AGENT)
    {
      switch(pPartLoadInfoOut-
>poAgentInfoList[i].oTableState)
      {
        case DB2LOADQUERY_NORMAL:
          fprintf(stderr,"%-8s\n", "NORMAL");
          break;

        case DB2LOADQUERY_UNCHANGED:
          fprintf(stderr,"%-8s\n", "UNCHANGED");
          break;

        case DB2LOADQUERY_LOAD_IN_PROGRESS:
          fprintf(stderr,"%-8s\n", "LOAD IN
PROGRESS");
          break;

        case DB2LOADQUERY_LOAD_PENDING:
          fprintf(stderr,"%-8s\n", "LOAD PENDING");
          break;

        default:
          fprintf(stderr,"%-8s\n", "UNKNOWN");
      }
    }
    else
    {
      fprintf(stderr,"%-8s\n", "N/A");
    }
  }

  /* Now print the partitioning statistics */
  fprintf(stderr,"\n");
  fprintf(stderr,"  Partitioning summary:\n");
  fprintf(stderr,"  Number of rows read        =
%d\n",
         (int)pPartLoadInfoOut->oRowsRdPartAgents);
  fprintf(stderr,"  Number of rows rejected    =
%d\n",
         (int)pPartLoadInfoOut->oRowsRejPartAgents);
  fprintf(stderr,"  Number of rows partitioned =
%d\n",
         (int)pPartLoadInfoOut->oRowsPartitioned);

  /* Now print the load statistics (i.e., number of
rows loaded, etc.) */
  fprintf(stderr,"\n");
  fprintf(stderr,"  Load summary:\n");
  fprintf(stderr,"  Number of rows read        =
%d\n",
         (int)pLoadInfoOut->oRowsRead);
  fprintf(stderr,"  Number of rows skipped     =
%d\n",
         (int)pLoadInfoOut->oRowsSkipped);
  fprintf(stderr,"  Number of rows loaded      =
%d\n",
         (int)pLoadInfoOut->oRowsLoaded);
  fprintf(stderr,"  Number of rows rejected    =
%d\n",
         (int)pLoadInfoOut->oRowsRejected);
  fprintf(stderr,"  Number of rows deleted     =
%d\n",
         (int)pLoadInfoOut->oRowsDeleted);
  fprintf(stderr,"  Number of rows committed   =
%d\n",
         (int)pLoadInfoOut->oRowsCommitted);
  fprintf(stderr,"\n");
}

/*ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
 ³ connect_to_TM - Connect to the transaction manager
database.
 ³
 ÀÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
 */
```

```
void connect_to_db_shared(char *in_dbname) {
   SQL_STRUCTURE sqlca  sqlca;
   char                *dbname_ptr;

   /* make sure we're in multi context mode */
   sqleSetTypeCtx(SQL_CTX_MULTI_MANUAL);

   /* create a context for this connection if there is
a context pointer */
   sqleBeginCtx(NULL, SQL_CTX_BEGIN_ALL, NULL,
&sqlca);

   if (in_dbname == NULL) {
      if ((dbname_ptr=getenv("TPCD_DBNAME")) != NULL)
          strcpy(UF_dbname, dbname_ptr);
   } else
      strcpy(UF_dbname, in_dbname);

   EXEC SQL CONNECT TO :UF_dbname IN SHARE MODE;

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

/*ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
 ³ disconnect_from_TM - Disconnect from the
transaction manager database.
 ³
 ÀÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
ÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄÄ
 */
void disconnect_from_db_shared() {
   EXEC SQL CONNECT RESET;

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


   /*  tpcd_load_staging uses db2Load api to load a
staging table */
int tpcd_load_staging(int staging_id     /*    what to
load           *
                                                specifi
ed by combination of
                                      TPCDBATCH_INSERT
or TPCDBATCH_DELETE +
                                      TPCDBATCH_ORDERS
or TPCDBATCH_LINEITEM
                                         */
             ,char    *dbname          /*    db name
*/
             ,int      updatePair      /*    update
pair number       */
             ,char    *flatfiles_path  /*    where
to read flat files */
             ,int      verbose         /*    verbose
option flag      */
                     )
/*  source for this program is based on sample in
tbload.sqc */
{
  SQL_API_RC rc = SQL_RC_OK;
  long sqlcode;
  long ix, jx, kx, tx;         /*  miscellaneous sizes
of things */
  db2LoadStruct loadParms;
  char pActionString[36] = "REPLACE INTO <table>";
  const char *pFileTypeModString = "COLDEL| FASTPARSE
ANYORDER";
  char msgflnm[256];
  sqlu_media_list loadMediaList;
  sqlu_location_entry inputLocationEntry;
  db2LoadIn loadInfoIn;
  db2LoadOut loadInfoOut;
  db2PartLoadIn partLoadInfoIn;
  db2PartLoadOut partLoadInfoOut;
  int num_agent_entries;   /* num agent entries for
partitioned load */

  /****   db2LoadNodeList partitioningDbPartNums;
```

```
***** not used */
  db2Uint16 mode;
  db2Uint16 isolatePartErrs;
  char *charptr;
  char *tblnmptr;
  char *msgflptr;
  char *dyn_mem_p;        /*  address of dynamically-
allocated memory   */
  /*    user can override conventional names for flat
file input by macro definitions as follows :  */
#if defined(FLAT_ORDNEW)
  char ordnew_name_flat[256] = FLAT_ORDNEW ;
  char ordnew_name_msg [260] = FLAT_ORDNEW ;
#endif
#if defined(FLAT_LINNEW)
  char linnew_name_flat[256] = FLAT_LINNEW ;
  char linnew_name_msg [260] = FLAT_LINNEW ;
#endif
#if defined(FLAT_ORDDEL)
  char orddel_name_flat[256] = FLAT_ORDDEL ;
  char orddel_name_msg [260] = FLAT_ORDDEL ;
#endif

  strncpy(UF_dbname, dbname, (sizeof(UF_dbname)-1));

  /* Connect to the database */
#ifdef SQLWINT
  connect_to_db_shared(UF_dbname);
  /*   connect_to_db_shared doesn't return if it fails
*/
  rc = 0;
#else
  EXEC SQL CONNECT TO :UF_dbname;
  rc = error_check();
#endif
  if (rc == 0)
  {

  /*************************************************
******************
   * Set up and initialize the db2Load API parameter
structure
   *************************************************
******************/
  memset(&loadParms, '\0', sizeof(db2LoadStruct));

  /* Set up the list of input source files. We are
using just one */
  /* which will be called "tbload.DEL"
*/
  loadParms.piSourceList = &loadMediaList;
  loadParms.piSourceList->media_type =
SQLU_SERVER_LOCATION;
  loadParms.piSourceList->sessions = 1;
  loadParms.piSourceList->target.location =
&inputLocationEntry;
  /*    choose input file name and staging table name
  **    different benchmarkers have adopted different
conventions for naming the flatfile input.
  *     This program has defaults for windows and non-
windows as shown below.
  **    user can override conventional names for flat
file input by macro definitions as shown
  **    in this case the message file name is formed
by suffixing input filename by .msg
  **    note that in all cases, the filename specified
here (as LOAD parameter) :
  **      . must contain a chunk number somewhere,
  **        represented by %d in the strings below and
substituted by chunk number by printf.
  **      . must NOT contain a partition number
anywhere, since LOAD adds this as a suffix,
  **
(zero-padded three-digit number)
  **    The names of the actual files must contain
both chunk number and partition suffix.
  */
  if (staging_id ==
(TPCDBATCH_INSERT+TPCDBATCH_ORDERS))
  {
        tblnmptr = "TPCDTEMP.ORDERS_NEW";
#if defined(FLAT_ORDNEW)
        charptr = ordnew_name_flat;
        msgflptr = ordnew_name_msg;
        *(msgflptr + sizeof(ordnew_name_msg) - 5) =
'\0';  /* stopper to ensure no overwrite in following
*/
        strcat(msgflptr,".msg");
#elif defined(SQLWINT)
        charptr = "order.tbl.new.u%d";
        msgflptr = "ord.msg.new.u%d";
#else
        charptr = "orders.tbl.u%d.new";
        msgflptr = "ord.msg.u%d.new";
#endif
  }
  else if (staging_id ==
(TPCDBATCH_INSERT+TPCDBATCH_LINEITEM))
  {
        tblnmptr = "TPCDTEMP.LINEITEM_NEW";
#if defined(FLAT_LINNEW)
        charptr = linnew_name_flat;
        msgflptr = linnew_name_msg;
        *(msgflptr + sizeof(linnew_name_msg) - 5) =
'\0';  /* stopper to ensure no overwrite in following
*/
        strcat(msgflptr,".msg");
#elif defined(SQLWINT)
        charptr = "lineitem.tbl.new.u%d";
        msgflptr = "lin.msg.new.u%d";
#else
        charptr = "lineitem.tbl.u%d.new";
        msgflptr = "lin.msg.u%d.new";
#endif
  }
  else
  {
        tblnmptr = "TPCDTEMP.ORDERS_DEL";
#if defined(FLAT_ORDDEL)
        charptr = orddel_name_flat;
        msgflptr = orddel_name_msg;
        *(msgflptr + sizeof(orddel_name_msg) - 5) =
'\0';  /* stopper to ensure no overwrite in following
*/
        strcat(msgflptr,".msg");
#elif defined(SQLWINT)
        charptr = "delete.new.%d";
        msgflptr = "rf2.msg.u%d.del";
#else
        charptr = "delete.%d.new";
        msgflptr = "rf2.msg.u%d.del";
#endif
  }
  sprintf(loadParms.piSourceList->target.location-
>location_entry, charptr, updatePair);
  strcpy(pActionString + 13, tblnmptr);        /*
table name  */
#ifdef TPCD_PLOAD_MESSAGES
  sprintf(msgflnm, msgflptr, updatePair);
#elif ! defined(SQLWINT)
  strcpy(msgflnm, "/dev/null"); /* apparently this
does suppress server msgs on unix/linux (maybe) */
#else  /*  SQLWINT and no messages */
  strcpy(msgflnm, "NUL"); /* apparently this does
suppress server msgs on windows (probably) */
#endif

  /*   allocate memory for all dynamic control blocks
*/
  /*   first compute total size   */
  ix = (sizeof(short) + strlen(pActionString) + 1);
/*  load action string */
  jx = (sizeof(short) + strlen(pFileTypeModString) +
1);  /*  file type modifier string */
  /*  for the agent info entries. In general, need to
know how many logical nodes in cluster -
  **  then allocate iMaxAgentInfoEntries = 3 * <number
of nodes>
  */
  if (   (charptr = getenv("TPCD_PHYS_NODE"))   /*
specified */
       && (num_agent_entries = atoi(charptr))    /*
non-zero  */
       && (charptr = getenv("TPCD_LN_PER_PN"))   /*
specified */
       && (tx = atoi(charptr))                   /*
non-zero  */
     )
       num_agent_entries *= tx;
/*  total number of logical nodes */
  else num_agent_entries = 0;
  if (num_agent_entries < 100)
```

```
    num_agent_entries = 100;
/*  minimum of 100 to be safe */
  else
  {
    num_agent_entries += 10;
/*    make it ...             */
    num_agent_entries *= 2;
/*    ... bigger              */
  }
  kx = (num_agent_entries *
(sizeof(db2LoadAgentInfo)));  /*  size required for
num_agent_entries agent structs  */
  tx = (ix + jx + kx);                          /*
total memory required for all control blocks */
  if (dyn_mem_p = malloc(tx))
  {
    /* Set up the load action string to "REPLACE INTO
<table>" */
    loadParms.piActionString = (struct sqlchar
*)dyn_mem_p;
    strcpy(loadParms.piActionString->data,
pActionString);
    loadParms.piActionString->length =
strlen(pActionString);

    /* Set the file type to DEL (i.e., an ASCII
delimited file) */
    loadParms.piFileType = (char *)SQL_DEL;

    /* Specify the ANYORDER file type modifier which
indicates to the  */
    /* load utility that it is not necessary to load
the rows of data  */
    /* into the table in the same order they appear in
the input file. */
    /* This can result in better load performance and
permits the use  */
    /* of multiple partitioning agents as well.
*/
      loadParms.piFileTypeMod = (struct sqlchar
*)(dyn_mem_p + ix);
    strcpy(loadParms.piFileTypeMod->data,
pFileTypeModString);
    loadParms.piFileTypeMod->length =
strlen(pFileTypeModString);

    /* Set up the name that will serve as a prefix for
the  */
    /* message files retrieved from each partition
that is  */
    /* participating in the load operation.
*/
    loadParms.piLocalMsgFileName = msgflnm;
    loadParms.piTempFilesPath = flatfiles_path;

    /* Set up and initialize the load input structure
*/
    memset(&loadInfoIn, '\0', sizeof(db2LoadIn));
    loadInfoIn.iNonrecoverable       =
SQLU_NON_RECOVERABLE_LOAD;
    loadInfoIn.iIndexingMode         =
SQLU_INX_AUTOSELECT;
    loadInfoIn.iAccessLevel          =
SQLU_ALLOW_NO_ACCESS;
    loadInfoIn.iLockWithForce        = SQLU_NO_FORCE;
    loadInfoIn.iCheckPending         =
SQLU_CHECK_PENDING_CASCADE_DEFERRED;
    loadInfoIn.iRestartphase         = ' ';
    loadInfoIn.iStatsOpt             =
SQLU_STATS_NONE;
    loadParms.piLoadInfoIn = (db2LoadIn *)&loadInfoIn;

    /* Set up and initialize the load output structure
*/
    memset(&loadInfoOut, '\0', sizeof(db2LoadOut));
    loadParms.poLoadInfoOut = (db2LoadOut
*)&loadInfoOut;

    /* Set up the callerac to indicate this is an
initial load operation */
    loadParms.iCallerAction = SQLU_INITIAL;

    /************************************************
********************
     * Set up the partitioning load input structure.
     *
     * NOTE: A value of NULL for any field in this
structure will
     *       result in the default value for the
option being used.
     ************************************************
********************/
    memset(&partLoadInfoIn, '\0',
sizeof(db2PartLoadIn));

    mode = DB2LOAD_LOAD_ONLY; /* input data already
partitioned and is loaded simultaneously on all
partitions */
    partLoadInfoIn.piMode = &mode;

    /*  partLoadInfoIn.piOutputNodes is already NULL :
load on all nodes */

    isolatePartErrs = DB2LOAD_NO_ISOLATION;  /*  any
error of any sort results in abort */
    partLoadInfoIn.piIsolatePartErrs =
&isolatePartErrs;

    partLoadInfoIn.piPartFileLocation =
flatfiles_path; /* input file path */

    loadParms.piPartLoadInfoIn = &partLoadInfoIn;

    /************************************************
********************
     * Set up the partitioned load output structure
     ************************************************
********************/
    memset(&partLoadInfoOut, '\0',
sizeof(db2PartLoadOut));

    /* Reserve space for 100 agent info entries. In
general, setting */
    /* iMaxAgentInfoEntries to 3 * <number of nodes>
in cluster      */
    /* should be sufficient.
*/
    partLoadInfoOut.iMaxAgentInfoEntries =
num_agent_entries;
    partLoadInfoOut.poAgentInfoList =
(db2LoadAgentInfo *)(dyn_mem_p + ix + jx);

    loadParms.poPartLoadInfoOut = &partLoadInfoOut;

    /************************************************
********************
     * Call db2Load
     ************************************************
********************/
    if (verbose)
    {
      fprintf(stderr,"\n  Load staging table \n"
                     "    action              : %s\n"
                     "    input data file name : %s\n"
                     "    temp file directory  : %s\n"
                     "    message file name    : %s\n"
        ,pActionString ,loadParms.piSourceList-
>target.location->location_entry
,loadParms.piTempFilesPath ,msgflnm);
    }

    /* refer to note earlier in ths file concerning
SQLZ_CURRENT_RELEASE */
#ifdef ALLDEBUG
    if(verbose) {
       printf("db2 load api called.\n");
       fflush(stdout);
    }
#endif
    rc = db2Load(
#ifdef SQLZ_CURRENT_RELEASE
        SQLZ_CURRENT_RELEASE
#else
        db2Version820
#endif
       ,&loadParms ,&sqlca);
    if(verbose) {
       printf("db2load api rc = %d \n", rc);
       fflush(stdout);
    }
    /* Display any warnings or errors */
    if (sqlca.sqlcode != 0)
```

```
    {
      rc = sqlcode = error_check();
    }
    else if (verbose)
    {
     /* Display a partition-level summary of the load
operation */
      PrintLoadSummary(loadParms.poLoadInfoOut,
                       loadParms.poPartLoadInfoOut,
                       &sqlca);
    }

    EXEC SQL COMMIT;
    sqlcode = error_check();
    if (rc == 0)
      rc = sqlcode;

    /* Free dynamically allocated memory */
    free(dyn_mem_p);
  }
  else
  {
    fprintf(stderr, "\ntpcd_load_staging parms %d %s
%d %s unable to allocate dynamic memory size %d\n"
                ,staging_id ,dbname ,updatePair
,flatfiles_path ,tx);
    LPRINTF(outstream, "\ntpcd_load_staging parms %d
%s %d %s unable to allocate dynamic memory size %d\n"
                ,staging_id ,dbname ,updatePair
,flatfiles_path ,tx);
  }

  /* Disconnect from the database */
#ifdef SQLWINT
  disconnect_from_db_shared();
#else
   EXEC SQL CONNECT RESET;
   sqlcode = error_check();
   if (rc == 0)
     rc = sqlcode;
#endif
  }
  return(rc);
}
```

tpcdbatch.h

```
/*
############################################################
######################### */
/* ## Licensed Materials - Property of IBM
*/
/* ##
*/
/* ## Governed under the terms of the International
*/
/* ## License Agreement for Non-Warranted Sample Code.
*/
/* ##
*/
/* ## (C) COPYRIGHT International Business Machines
Corp. 1990 - 2005            */
/* ## All Rights Reserved.
*/
/* ##
*/
/* ## US Government Users Restricted Rights - Use,
duplication or                 */
/* ## disclosure restricted by GSA ADP Schedule
Contract with IBM Corp.          */
/*
############################################################
######################### */
/**************************************************
************************
*
*   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
*   3 mar 05 jel  Increased version to 5.8
**************************************************
************************/

/** 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>
#include <sys/shm.h>
#include <sys/wait.h>
#else
#include <windows.h>
#include <sys\timeb.h>
#endif
#include <errno.h>
#include <signal.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 "5.8"

#define TPCDBATCH_NONSQL      10
/* @d23684 tjg */
#define TPCDBATCH_SELECT      20
#define TPCDBATCH_OPT_DRIVER  25       /*  .opt driver
command jel  */
#define TPCDBATCH_NONSELECT   30
#define TPCDBATCH_EOBLOCK     40
/* @d30369 tjg */
#define TPCDBATCH_INSERT      50
#define TPCDBATCH_DELETE      60
#define TPCDBATCH_ORDERS      01
#define TPCDBATCH_LINEITEM    02
   /*  tpcd_load_staging uses db2Load api to load a
staging table */
int tpcd_load_staging(int staging_id     /*    what to
load
```

```
                                                specifi
ed by combination of
                                    TPCDBATCH_INSERT
or TPCDBATCH_DELETE +
                                    TPCDBATCH_ORDERS
or TPCDBATCH_LINEITEM
                                         */
              ,char *dbname              /*    db name
*/
              ,int updatePair            /*    update
pair number     */
              ,char *flatfiles_path   /* where to read
flat files      */
              ,int      verbose          /*    verbose
option flag     */
                    );

#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 **/

extern int LPRINTF(FILE *stream, char *va_alist, ...);
/* write query output to either buffer if it exists,
else directly to file */

/** 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   */
```

makefile.sun

```
#########################################################
###############
# 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.
#  NOTE:  This makefile is for pe,  use Makefile.pe
for PE (until I
#    can figure out a way to set the
"TPCD_NONPARTITIONED" define for the compile
#########################################################
###############

TPCD_DBNAME=tpcdqual
#LOCAL=tpcd

BASE=$(HOME)/sqllib
# if using an installed db2 image use the 2nd
link_flags value
LINK_FLAGS= -o $@ -L$(BASE)/lib
LIB= -ldb2 -lm
COMPILER=cc
LIB_LINKER=ld
```

```
EXTRA_CFLAGS= -m64
COMPILE_FLAGS=$(EXTRA_CFLAGS) -DTPCD_PLOAD_SCRIPTS
-DSQLSUN -DSQLAIX -DSQLUNIX -DNO_PARALLEL_FORMAT_FETCH
-D_HAVE_STDARG -c -I$(BASE)/include
#COMPILE_FLAGS=$(EXTRA_CFLAGS) -DSQLUNIX -c -I
$(BASE)/include
LIB=-L$(BASE)/lib -R$(BASE)/lib -ldb2 -lm -L. -m64

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 '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 'connect to $(TPCD_DBNAME) \n prep
tpcdUF.sqc BINDFILE PACKAGE ISOLATION RS BLOCKING ALL
OPTLEVEL 1 DEGREE 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 $(LIB)
```

Appendix F. Pricing information

For IBM DB2 pricing please contact:

Richard Hughes
212-493-2065
rhughes@us.ibm.com

For Sun pricing please contact:

Daryl Madura
503-617-8588
daryl.madura@sun.com