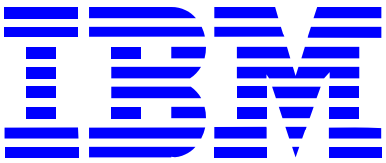


TPC Benchmark™ H Full Disclosure Report

IBM @server p5 575

Using

IBM DB2 Universal Database 8.2

First Edition

May 20, 2005

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

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

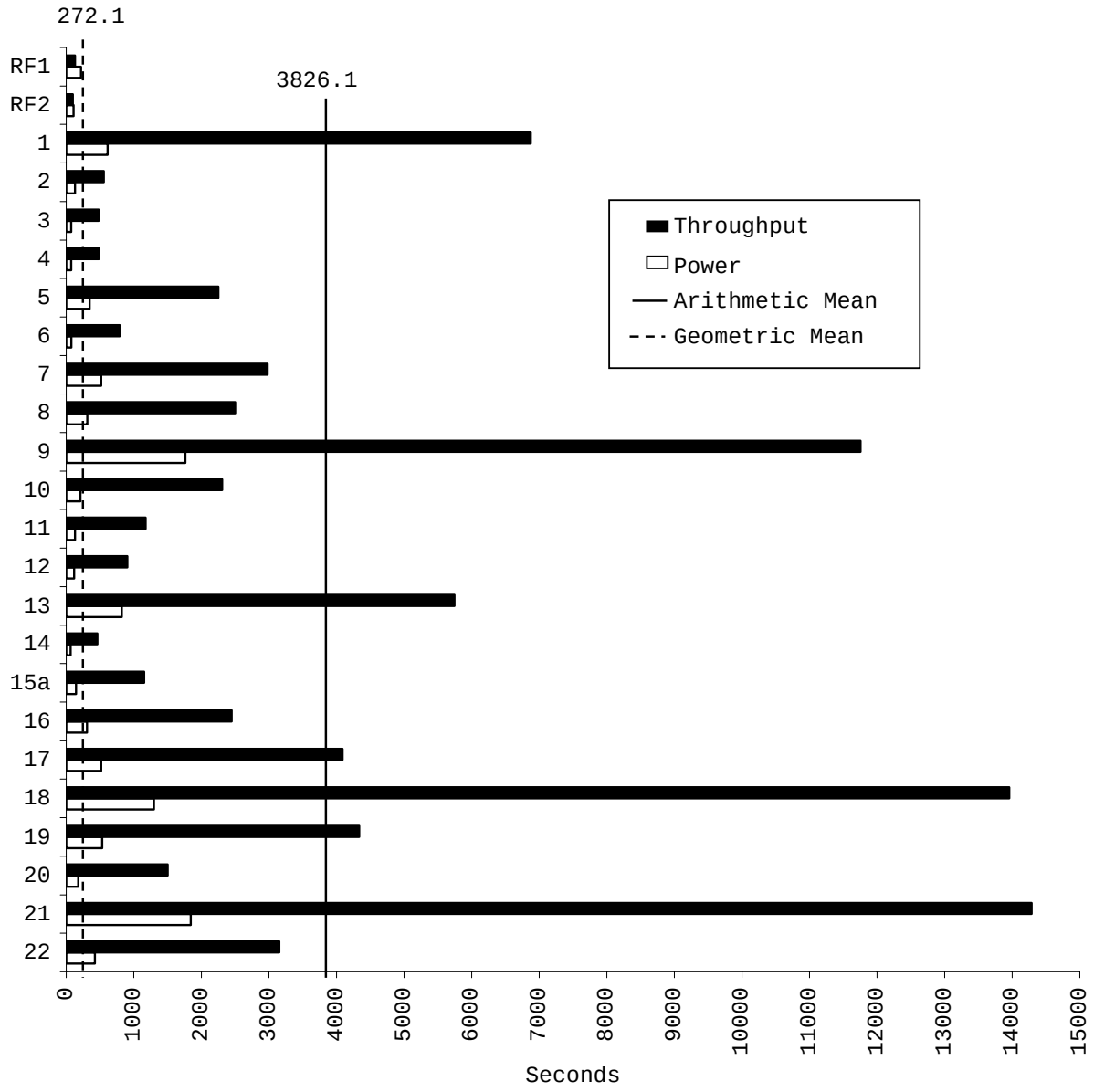
Trademark of International Business Machines Corporation:
IBM
the IBM logo
@server
pSeries
POWER5
TotalStorage
AIX
AIX 5L
DB2
DB2 Universal Database

First Edition May 20, 2005

IBM	**IBM @server p5 575 with DB2 UDB 8.2**	**TPC-H Rev. 2.1.0** **Report Date: May 20, 2005**

Total System Cost	Composite Query per Hour Rating	Price/Performance
6,367,595 USD	104,100.1 QphH@10000GB	61 USD Price/QphH@10000GB

Database Size	Database Manager	Operating System	Other Software	Availability Date
10000GB	DB2 UDB 8.2	AIX 5L V5.3	None	August 15, 2005



Database Load Time: 6:35:14	Load included backup: N	Total Data Storage/Database Size: 6.65
RAID (base tables): Y	RAID (Base Tables and Auxiliary Data Structures): Y	RAID (All): N

System Configurations

8 IBM @server p5 575 servers, each server with

Processors	8 x 1900MHz POWER5 with 8 x 36MB L3 Cache
Memory	128GB
Disk Controllers	12 x 2Gigabit Fibre Channel PCI-X Adapter & 3 x IBM TotalStorage DS4500 dual controllers
Disk Drives	240 x 36.4GB 15K 2Gb FC drives ; 2 x 73.4GB 15K Ultra4 SCSI drives, with 4 additional on server1
Network	2 Integrated dual Gigabit Ethernet
Total Disk Storage	66,455.18 GB (GB is defined as 1024 * 1024 * 1024 bytes)

Description	Part No.	Source	Unit Price	Qty	Ext Price	Maint Price
Server Hardware						
Server 9118 Model 575	9118-575	1	4,565	8	36,520	42,720
RIO-2(Remote I/O-2)Cbl, 1.2M	3146	1	420	16	6,720	
73.4 GB 10,000 RPM Ultra320 SCSI Disk Drive	3274	1	900	20	18,000	
Pwr.Cbl.Grp, 09U(or 09U Lft)	3800	1	75	8	600	
8192MB (4x2048MB) DIMMs, 208-pin, 8NS	4454	1	17,695	128	2,264,960	
2 Gigabit Fibre Channel PCI-X Adapter	5716	1	2,720	96	261,120	
System Rack	5793	1	10,000	2	20,000	16,800
I/O Drawer, 20 Slots, 8 Disk Bays	5794	1	30,000	8	240,000	31,680
Bulk Power Regulator	6186	1	4,000	12	48,000	
Slim Line Doors	6251	1	6,000	2	12,000	
Weight Distribution Plate	6855	1	750	2	1,500	
I/O Assembly with PCI-X and RIO-2 capability	7210	1	4,300	8	34,400	
Ethernet Cable, 15M, HMC Console to System	7802	1	31	2	62	
Bulk Power Controller Assembly	7803	1	4,000	4	16,000	
8-Way 1.9GHz POWER5 Processor, 288 MB L3	7836	1	40,000	8	320,000	70,656
Bulk Power Distribution Assembly	7837	1	2,500	4	10,000	
Power Cables, 4x, 05U	7854	1	650	8	5,200	
Ethernet Cables, Node to Hubs, EIA01 through EIA17	7920	1	100	8	800	
RIO-2(Remote I/O-2)Cbl, 0.6M	7924	1	300	8	2,400	
Line Cord, 6AWG/Type W, 14ft, IEC309 60A	8688	1	1,500	4	6,000	
HMC 1:7310-C03 Desktop Hardw.Mgmt.Console	7310-C03	1	1,830	1	1,830	1,152
IBM ThinkVision C170 17-inch Color Monitor	3631	1	250	1	250	
10/100 Mbps Ethernet PCI Adapter II	4962	1	412	1	412	
Power Cord (6-foot), To Wall (125V, 15A),	6470	1	14	2	28	
Ethernet Cable, 6M, Hardware Management	7801	1	12	1	12	
				Subtotal	**3,306,814**	**163,008**
Storage						
DS4500 Disk System	1742-90U	1	49,900	24	1,197,600	
DS4000 EXP710 Storage Expansion	1740-710	1	6,000	144	864,000	
2Gb FC, 36.4GB/15K Drive	5212	1	1,115	1,920	2,140,800	
Short Wave SFP	2210	1	499	672	335,328	
Fiber Cable 25m	5625	1	189	96	18,144	
Fiber Cable 1m	5601	1	79	288	22,752	
2Gb Mini HUB	3507	1	899	96	86,304	
NetBay42 Standard Rack	9307-4SX	1	1,489	14	20,846	4,200
3 Year Warranty Service Upgrade 1740-1RU 24x7x4	1740-710W24	1	760	144		109,440
3 Year Warranty Service Upgrade 1742-90U 24x7x4	1742-90UW24	1	1,087	24		26,088
				Subtotal	**4,685,774**	**139,728**

Server Software						
AIX 5.3 (media only)	5962-A5L	1	50	1	50	
AIX Software	5765-G03	1	1,225	64	78,400	
AIX Software Maintanance (1Y)	5771-496	1	725	192		139,200
AIX Software Maintanance 24x7 Upgrade (1Y)	5771-498	1	183	192		35,136
HMC Software Support 1 Year	5771-612	1	250	3		750
HMC Software Support 24x7 1 Year	5771-614	1	87	3		261
C for AIX user Lic+SW maint 12 MO	D5A1DLL	1	515	1	515	
C for AIX user annual SW maint renewal	E1A1FLL	1	103	2		206
DB2 UDB ESE License/1-yr. Maint.		1	22,608	64	1,446,912	
DB2 UDB ESE Support 1Yr/Proc		1	1,077	128		137,856
DB2 UDB ESE DPF License/1-yr. Maint.		1	6,791	64	434,624	
DB2 UDB ESE DPF Support 1Yr/Proc		1	323	128		41,344
				Subtotal	**1,960,501**	**354,753**
Third Party Hardware/Software						
SMC 16 Port 10/100/1000 Layer 2 Gigabit Switch	587912	2	312	4	1,248	
				Subtotal	**1,248**	
				Total	**9,954,337**	**657,489**
					IBM Total System Discounts*	**-4,244,231**
					Three-Year Cost of Ownership	**6,367,595**
					QphH	**104,100.1**
					$/QphH	**61**

*Discounts are based on US list prices for similar quantities & configurations

Pricing Sources:

1 IBM: Bill Casey, eServer pSeries Offering Manager, wrcasey@us.ibm.com, 1-512-838-1422

1 IBM DB2: Bernard Spang, Director, Database Market Management, spang@us.ibm.com, 1-914-766-1491

2: CDW.COM

Audited by: Francois Raab of Infosizing (www.infosizing.com)

The system as configured for the test will be available August 15, 2005.

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

Numerical Quantities Summary

Measurement Results

Database Scaling (SF/Size)	=	10000
Total Data Storage/Database Size	=	6.65
Start of Database Load	=	5/11/2005 2:58:04
End of Database Load	=	5/11/2005 9:33:18
Database Load Time	=	6:35:14
Query Streams for Throughput Test	=	9
TPC-H Power Metric (QppH@10000GB)	=	132,295.0
TPC-H Throughput Metric (QthH@10000GB)	=	81,914.1
Composite Query-per-Hour Rating (QphH@10000GB)	=	104,100.1
Total System Price over 3 years	=	6,367,595 USD
TPC-H Price/Performance Metric ($/QphH@10000GB)	=	61 USD

Measurement Intervals

Measurement Interval in Throughput Test (Ts) = 87,018 seconds

Duration of Stream Execution

Date and Time Stamps								Duration
Stream Id	Seed Used	Query		RF1 Start		RF2 Start		Queries
		Query End		RF1 End		RF2 End		RFs
00	511093318	5/ 11/ 2005	15:10:35	5/ 11/ 2005	15:06:57	5/ 11/ 2005	18:05:13	2:54:38
		5/ 11/ 2005	18:05:13	5/ 11/ 2005	15:10:35	5/ 11/ 2005	18:07:03	0:05:28
01	511093319	5/ 11/ 2005	18:07:28	5/ 12/ 2005	17:43:48	5/ 12/ 2005	17:46:35	23:25:28
		5/ 12/ 2005	17:32:56	5/ 12/ 2005	17:46:35	5/ 12/ 2005	17:48:04	0:04:16
02	511093320	5/ 11/ 2005	18:07:28	5/ 12/ 2005	17:48:04	5/ 12/ 2005	17:50:12	23:06:34
		5/ 12/ 2005	17:14:02	5/ 12/ 2005	17:50:12	5/ 12/ 2005	17:51:44	0:03:40
03	511093321	5/ 11/ 2005	18:07:28	5/ 12/ 2005	17:51:44	5/ 12/ 2005	17:53:51	23:33:42
		5/ 12/ 2005	17:41:10	5/ 12/ 2005	17:53:51	5/ 12/ 2005	17:55:24	0:03:40
04	511093322	5/ 11/ 2005	18:07:28	5/ 12/ 2005	17:55:24	5/ 12/ 2005	17:57:32	23:29:58
		5/ 12/ 2005	17:37:27	5/ 12/ 2005	17:57:32	5/ 12/ 2005	17:59:16	0:03:52
05	511093323	5/ 11/ 2005	18:07:28	5/ 12/ 2005	17:59:16	5/ 12/ 2005	18:01:28	23:23:50
		5/ 12/ 2005	17:31:18	5/ 12/ 2005	18:01:28	5/ 12/ 2005	18:03:01	0:03:45
06	511093324	5/ 11/ 2005	18:07:28	5/ 12/ 2005	18:03:01	5/ 12/ 2005	18:05:09	23:36:20
		5/ 12/ 2005	17:43:48	5/ 12/ 2005	18:05:09	5/ 12/ 2005	18:06:43	0:03:42
07	511093325	5/ 11/ 2005	18:07:28	5/ 12/ 2005	18:06:43	5/ 12/ 2005	18:08:43	23:03:58
		5/ 12/ 2005	17:11:26	5/ 12/ 2005	18:08:43	5/ 12/ 2005	18:10:23	0:03:40
08	511093326	5/ 11/ 2005	18:07:28	5/ 12/ 2005	18:10:23	5/ 12/ 2005	18:12:25	23:13:27
		5/ 12/ 2005	17:20:55	5/ 12/ 2005	18:12:25	5/ 12/ 2005	18:14:08	0:03:45
09	511093327	5/ 11/ 2005	18:07:28	5/ 12/ 2005	18:14:08	5/ 12/ 2005	18:16:11	23:32:44
		5/ 12/ 2005	17:40:13	5/ 12/ 2005	18:16:11	5/ 12/ 2005	18:17:46	0:03:38

Timing Interval (in seconds)

Stream ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
Stream 0	612.3	127.3	71.6	73.1	346.5	76.8	515.9	312.1	1761.5	210.5	128.5	116.4
Stream 01	7126.5	577.7	574.4	255.2	2222.4	2936.4	2648.0	2342.4	11364.4	1600.8	1142.0	966.7
Stream 02	6769.0	685.8	581.9	597.5	1980.9	667.2	2782.6	2584.3	12347.1	1583.8	1030.8	912.6
Stream 03	7093.3	377.0	164.7	548.2	2113.3	432.6	2840.9	2529.8	11313.8	1536.0	2136.4	418.1
Stream 04	7262.0	484.8	331.6	504.9	2760.4	540.6	3083.2	2335.5	11765.1	2557.8	1152.3	1077.2
Stream 05	6603.8	479.2	483.9	554.4	2199.2	397.9	3157.5	2461.1	12866.3	1751.4	1099.2	1078.4
Stream 06	6686.8	801.6	589.1	539.5	2092.2	735.1	2945.6	2615.1	13563.4	1679.9	1059.0	855.6
Stream 07	7228.3	471.1	541.9	448.5	2207.7	498.8	3033.8	2284.8	11237.4	1813.6	1056.2	908.1
Stream 08	6616.6	490.5	484.8	459.2	2580.6	401.3	3422.9	2811.3	10679.5	1598.8	726.7	939.7
Stream 09	6473.0	650.1	550.4	453.5	2104.0	487.7	2911.4	2532.5	10664.2	6622.3	1142.2	989.0
Minimum	6473.0	377.0	164.7	255.2	1980.9	397.9	2648.0	2284.8	10664.2	1536.0	726.7	418.1
Average	6873.3	557.5	478.1	484.5	2251.2	788.6	2980.7	2499.6	11755.7	2304.9	1171.6	905.0
Maximum	7262.0	801.6	589.1	597.5	2760.4	2936.4	3422.9	2811.3	13563.4	6622.3	2136.4	1078.4

Stream ID	Q13	Q14	Q15a	Q16	Q17	Q18	Q19	Q20	Q21	Q22	RF1	RF2
Stream 0	823.7	65.0	146.1	306.1	515.1	1297.2	530.7	177.5	1842.4	421.4	218.2	109.8
Stream 01	5204.2	445.1	1119.9	2549.2	3410.7	14107.5	4123.3	1573.0	14644.8	3393.2	166.8	88.6
Stream 02	5653.4	461.2	1112.5	2499.3	4087.3	12709.9	4669.1	1704.0	14965.8	2807.9	128.3	91.5
Stream 03	6567.0	455.0	977.2	2204.8	5470.3	14268.6	4249.0	1551.4	14990.8	2584.2	127.2	93.1
Stream 04	6380.7	538.2	1303.7	2536.4	4480.7	12925.7	4852.8	641.2	14119.5	2963.9	127.7	103.9
Stream 05	5508.8	447.3	1141.4	2931.8	4106.6	13143.5	4779.5	1579.6	14528.4	2930.9	132.6	92.9
Stream 06	5626.7	432.9	1179.0	2385.3	3919.1	17616.1	4064.2	1689.3	11021.4	2883.0	127.5	94.4
Stream 07	5729.5	429.5	1255.1	2926.8	3744.9	12908.5	4308.2	1366.0	15325.4	3313.8	120.5	99.4
Stream 08	5217.6	427.5	1185.3	2505.5	3623.2	14941.4	4229.2	1692.4	13977.1	4595.5	122.3	102.7
Stream 09	5810.4	493.2	1107.3	1486.2	3972.1	12970.7	3722.8	1714.6	14987.2	2919.3	123.4	94.6
Minimum	5204.2	427.5	977.2	1486.2	3410.7	12709.9	3722.8	641.2	11021.4	2584.2	120.5	88.6
Average	5744.3	458.9	1153.5	2447.3	4090.5	13954.7	4333.1	1501.3	14284.5	3154.6	130.7	95.7
Maximum	6567.0	538.2	1303.7	2931.8	5470.3	17616.1	4852.8	1714.6	15325.4	4595.5	166.8	103.9

INFO SIZING

TPC TRANSACTION PROCESSING PERFORMANCE COUNCIL CERTIFIED AUDITOR

Benchmark Sponsors: Haider Rizvi
IBM Canada Ltd;
8200 Warden Avenue
Markham, Ontario L6G 1C7

John J. Makis
IBM eServer Performance
11501 Burnet Road
Austin, TX 78758

May 18, 2005

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

Platform: **IBM @server® p5 575, 8-node cluster**
Database Manager: **IBM DB2 UDB 8.2**
Operating System: **AIX 5L V5.3**

The results were:

CPU (Speed)	Memory	Disks	QphH@10,000GB
Eight (8) **IBM @server® p5 575** (each with)			
8 x POWER5 (1.9 GHz)	8x 36 MB L3 Cache 128 GB Main	240 x 36.4 GB FC 2 x 73.4 GB Ultra4	**104,100.1**

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:

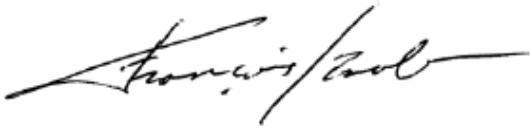
- The database records were defined with the proper layout and size
- The database population was generated using DBGEN
- The database was properly scaled to 10,000GB and populated accordingly
- The compliance of the database auxiliary data structures was verified
- The database load time was correctly 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 an approved query variant
- The execution of the queries against the SF1 database produced compliant answers
- A compliant implementation specific layer was used to drive the tests
- The throughput tests involved 9 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

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

Abstract

This report documents the full disclosure information required by the TPC Benchmark™ H Standard Specification Revision 2.1.0 dated August 14, 2003 for measurements on the IBM @server p5 575.

The software used includes AIX 5L V5.3 operating system with DB2 Universal Database 8.2.

Preface

TPC Benchmark™ H Standard Specification was developed by the Transaction Processing Performance Council (TPC). It was released on February 26, 1999, and most recently revised (Revision 2.1.0) on August 14, 2003. This is the full disclosure report for benchmark testing of the IBM @server p5 575 according to the TPC Benchmark™ H Standard Specification.

TPC Benchmark™ H is a Decision Support benchmark. It is a suite of business oriented queries and concurrent updates. The queries and the data populating the database have been chosen to have broad industry-wide relevance while maintaining a sufficient degree of ease of implementation. 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 (e.g., via a point and click GUI interface);
- Are far more complex than most OLTP transactions;
- Include a rich breadth of operators and selectivity constraints;
- Generate intensive activity on the part of the database server component of the system under test;
- Are executed against a database complying to specific population and scaling requirements;
- Are implemented with constraints derived from staying closely synchronized with an on-line production database.

The TPC-H operations are modeled as follows:

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

- To achieve the optimal compromise between performance and operational requirements, the database administrator can set, once and for all, the locking levels and the concurrent scheduling rules for queries and refresh functions.

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

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

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

TPC-H uses terminology and metrics that are similar to other benchmarks, originated by the TPC and others. Such similarity in terminology does not in any way imply that TPC-H results are comparable to other benchmarks. The only benchmark results comparable to TPC-H are other TPC-H results compliant with the same revision.

Despite the fact that this benchmark offers a rich environment representative of many decision support systems, this benchmark does not reflect the entire range of decision support requirements. In addition, the extent to which a customer can achieve the results reported by a vendor is highly dependent on how closely TPC-H approximates the customer application. The relative performance of systems derived from this benchmark does not necessarily hold for other workloads or environments. Extrapolations to any other environment are not recommended.

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

Benchmark sponsors are permitted several possible system designs, provided that they adhere to the model described in Clause 6. A full disclosure report (FDR) of the implementation details, as specified in Clause 8, must be made available along with the reported results.

1.0 General Items

1.1. Benchmark Sponsor

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

This benchmark was sponsored by International Business Machines Corporation.

1.2. Parameter Settings

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

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

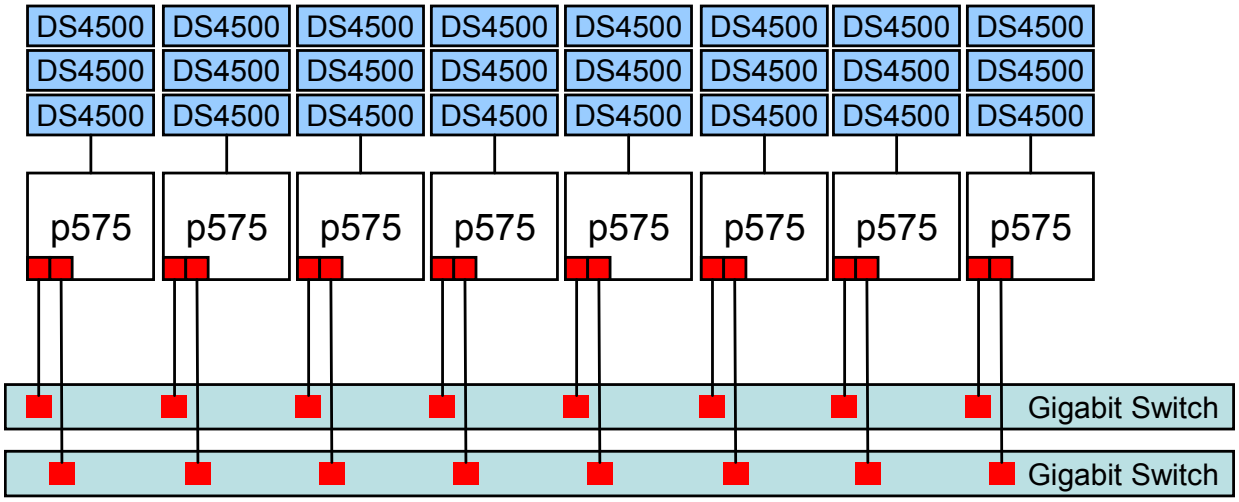
Appendix A "Tunable Parameters" contains a list of all DB2 parameters and operating system parameters. Session initialization parameters can be set during or immediately after establishing the connection to the database within the tpcdbatch program documented in Appendix D "Driver Source Code". This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

1.3. Configuration Diagrams

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

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

IBM @server p5 575 Benchmark Configuration



The system was a 8 node cluster of IBM @server p5 575 systems each with

- 8 POWER5 1900MHz processors with 8 x 36MB L3 cache
- 128 GB of memory
- 12 IBM 2Gb Fibre Channel PCI-X Adapters
- 2 73.4 GB 15K RPM Internal disk drives with 4 additional drives on node1
- 240 36.4 GB 15K RPM external disk drives
- 18 EXP710 disk enclosures
- 3 IBM TotalStorage DS4500 dual controllers
- 2 Integrated dual Gigabit Ethernet

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

2.0 Clause 1: Logical Database Design Related Items

Appendix B "Database Build Scripts" contains the programs and input files used to load the test database. The test and qualification databases use the same table definitions, indices and partitioning methods. Thus, the buildtpcd script documented in Appendix B was used for both the qualification and test databases except that different input files were used to define the tablespace devices and sizes.

2.1. Table Definitions

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

Appendix B "Database Build Scripts" contains the table definitions and the programs to load the database.

2.2. Database 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.

Appendix B "Database Build Scripts" contains the DDL for the table and index definitions.

2.3. Horizontal Partitioning

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

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

2.4. Replication

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

No replication was used.

3.0 Clause 2: Queries and Refresh Functions

3.1. Query Language

The query language used to implement the queries must be identified (e.g.,"RALF/SQL-Plus").

SQL was the query language used.

3.2. Verification for the Random Number Generator

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

The supplied QGEN version 2.1.1b and DBGEN 2.1.1b were used.

3.3. Substitution Parameters

3.3.1. Method of Generation

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

The supplied QGEN version 2.1.1b was used to generate the substitution parameters.

3.4. Query Text

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

Appendix C.1 "Qualification Query Output" contains the output for each of the queries.

Query variant Q15a was used in this disclosure.

The functional query definitions and variants used in this disclosure use the following minor query modifications.

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

3.5. Query Substitution Parameters and Seeds

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

Appendix C3, "Query Substitution Parameters" contains the query substitution parameters used in the performance tests.

3.6. Isolation Level

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

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

3.7. Refresh Functions

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

The refresh functions are part of the implementation specific layer/driver code included in Appendix D "Driver Source Code".

4.0 Clause 3: Database System Properties

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

All ACID tests were conducted according to specification. The Atomicity, Isolation, Consistency and Durability tests were performed on the @server p5 575. Appendix E. "Acid Transaction Source Code" contains the source code for the ACID transaction and query.

4.1. Atomicity Requirements

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

4.1.1. Atomicity of Completed Transaction

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

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

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

4.1.2. Atomicity of Aborted Transactions

Perform the ACID 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 ORDER, LINEITEM, and HISTORY tables.

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

1. The total price from the ORDER table and the extended price from the LINEITEM table were retrieved for a random Orderkey. The number of records in the HISTORY table was also retrieved.
2. The ACID transaction was executed for the Orderkey used in step 1.
3. The transaction was rolled back.
4. The total price and the extended price were retrieved for the same orderkey used in step 1 and step 2. It was verified that the extended price and the total price were the same as in step 1. The

number of records in the HISTORY table was retrieved again and verified to be the same as in step1.

4.2. Consistency Requirements

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

4.2.1. Consistency Condition

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

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

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

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

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

4.2.2. Consistency Tests

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

The queries defined in 4.2.1 , "Consistency Condition" were run after initial database build and prior to executing the ACID transaction. The queries showed that the database is in a consistent state.

After executing 10 streams of 100 ACID transactions each, the queries defined in 4.2.1 , "Consistency Condition" were run again. The queries showed that the database was still in a consistent state.

4.3. Isolation Requirements

4.3.1. Isolation Test 1

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

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

1. 1st session: Start an ACID transaction with a randomly selected O_KEY,L_KEY and DELTA. The transaction is delayed for 60 seconds just prior to the Commit.
2. 2nd session: Start an ACID query for the same O_KEY as in the ACID transaction.

3. 2nd session: The ACID query attempts to read the file but is locked out by the ACID transaction waiting to complete.
4. 1st session: The ACID transaction is released and the Commit is executed releasing the record. With the LINEITEM record now released, the ACID query can now complete.
5. 2nd session: Verify that the ACID query delays for approximately 60 seconds and that the results displayed for the ACID query match the input for the ACID transaction.

4.3.2. Isolation Test 2

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

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

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

4.3.3. Isolation Test 3

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

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

1. 1st session: Start an ACID transaction T1 for a randomly selected O_KEY, L_KEY and DELTA. The transaction is delayed for 60 seconds just prior to the COMMIT.
2. 2nd session: Start a second ACID transaction T2 for the same O_KEY, L_KEY, and for a randomly selected DELTA2. This transaction is forced to wait while the 1st session holds a lock on the LINEITEM record requested by the second session.
3. 1st session: The ACID transaction T1 is released and the Commit is executed, releasing the record. With the LINEITEM record now released, the ACID transaction T2 can now complete.
4. Verify that:

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

4.3.4. Isolation Test 4

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

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

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

4.3.5. Isolation Test 5

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

1. 1st session: Start an ACID transaction, T1, for a randomly selected O_KEY, L_KEY and DELTA. The ACID transaction was suspended prior to COMMIT.
2. 2nd session: Start a second ACID transaction, T2, which selects random values of PS_PARTKEY and PS_SUPPKEY and returns all columns of the PARTSUPP table for which PS_PARTKEY and PS_SUPPKEY are equal to the selected values.
3. T2 completed.
4. T1 was allowed to complete.
5. It was verified that the appropriate rows in the ORDERS, LINEITEM and HISTORY tables have been changed.

4.3.6. Isolation Test 6

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

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

4.4. Durability Requirements

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

4.4.1. Permanent Failure of Single Durable Medium and Loss of System Power

These tests were combined and conducted on the qualification database. The following steps were performed:

1. The consistency test described in section 4.2.1 was verified.
2. The current count of the total number of records in the HISTORY table was determined giving hist1.
3. A test to run 200 ACID transactions on each of 10 execution streams was started such that each stream executes a different set of transactions.
4. One of the disks containing the DB2 transaction log recovery data, database table data, and database index was removed from the enclosure after at least 20 ACID transactions had completed from each of the execution streams.
5. Because the disks were in RAID 5 configuration the applications continued running the ACID transactions.
6. The system was shutdown by switching off the power for all system components, after at least a total of 100 transactions had completed for each stream.
7. The system was powered back on and rebooted, and the database was restarted.
8. Step 2 was performed giving hist2. It was verified that hist2 - hist1 was greater than or equal to the number of records in the success file.
9. Consistency condition described in 4.2.1 was verified.

4.4.2. Loss of Switch Power

This test was conducted on the qualification database. The following steps were performed:

1. The consistency test described in section 4.2.1 was verified.
2. The current count of the total number of records in the HISTORY table was determined giving hist1.
3. A test to run 200 ACID transactions on each of 10 execution streams was started such that each stream executes a different set of transactions.
4. The Gigabit switch was disconnected from the system.
5. Database detected the network loss and terminated processing
6. Network connections were reestablished and the database was restarted
7. Step 2 was performed giving hist2. It was verified that hist2 - hist1 was greater than or equal to the number of records in the success file.

8. Consistency condition described in 4.2.1 was verified.

4.4.3. Failure of Storage Controller, and Loss of System Power

This test was conducted on the qualification database. The following steps were performed:

1. The consistency test described in section 4.2.1 was verified.
2. The current count of the total number of records in the HISTORY table was determined giving hist1.
3. A test to run 200 ACID transactions on each of 10 execution streams was started such that each stream executes a different set of transactions.
4. A controller which was on the preferred path of database log data, database table data, and database index was removed after at least 20 ACID transactions had completed from each of the execution streams.
5. The storage system automatically switched the disks to the backup disk controller. The applications continued running the ACID transactions.
6. The system was shutdown by switching off the power for all system components, after at least a total of 100 transactions had completed for each stream.
7. The system was powered back on and rebooted, and the database was restarted.
8. Step 2 was performed giving hist2. It was verified that hist2 - hist1 was greater than or equal to the number of records in the success file.
9. Consistency condition described in 4.2.1 was verified.

5.0 Clause 4: Scaling and Database Population Related Items

5.1. Cardinality of Tables

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

The following table contains the TPC Benchmark™ H defined tables and the number of rows for each table as they existed upon build completion:

Table	Rows
Lineitem	59,999,994,267
Orders	15,000,000,000
Customer	1,500,000,000
Supplier	100,000,000
Part	2,000,000,000
Partsupp	8,000,000,000
Nation	25
Region	5

5.2. Distribution of Tables and Logs

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

DB2 was configured on an eight node cluster of IBM @server p5 575 servers. Each node had:

- 12 IBM 2Gb Fibre Channel PCI-CR adapters
- 240 36.4GB external disk drives.
- Node 1 had 6 73.4GB internal drives and nodes 2 - 8 each had 2 73.4GB drives

Each IBM @server p5 575 had three IBM TotalStorage DS4500 dual controllers. Each of the two controllers in each DS4500 had eight 3+p RAID 5 arrays and eight non-RAID disks. Each of the controllers was accessed through one FC adapter.

Permanent tables, their auxiliary data structures, and database logs resided in tablespaces created on the RAID 5 arrays. Temporary tables resided in tablespaces created on the non-RAID disks. See Appendix B "Database Build Scripts".

The Operating System resided on an internal disk on each node. The database software resided on an internal disk on each node. Four additional disks on node 1 were used for the benchmark executable programs and results. This filesystem was NFS mounted by nodes 2 - 8, from node 1.

5.3. Mapping of Database Partitions/Replications

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

The database was not replicated. The database was logically partitioned into 128 logical data partitions, 16 data partitions on each physical server.

5.4. Implementation of RAID

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

RAID level 5 was used for database tables, indexes, and recovery logs.

5.5. DBGEN Modifications

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

The standard distribution of DBGEN version 2.1.1b was used.

5.6. Database Loading

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

The database load time was 6:35:14.

5.7. Data Storage Ratio

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

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

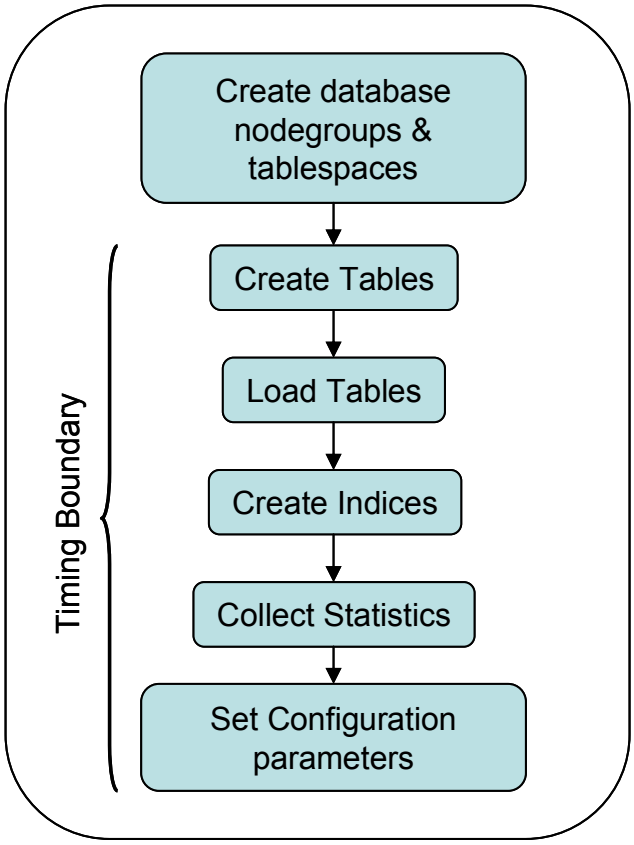
Disk Type	Number of Disks	Space per Disk	Sub-Total Disk Space	Database Size	Data Storage Ratio
ULTRA4 73.4GB	20	68.359 GB	1,367.18 GB		
2GB FC 36.4GB	1920	33.900 GB	65,088.00 GB		
Total			66,455.18 GB	10,000GB	6.65

5.8. Details of Database Loading

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

Flat files for each of the tables were created using DBGEN. Appendix B "Database Build Scripts" contains the programs and input files used to load the database.

Database Load Procedure:



6.0 Clause 5: Performance Metrics and Execution-Rules Related Items

6.1. System Activity between Load and Performance Tests

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

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

6.2. Steps in the Power Test

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

The following steps were used to implement the power test:

1. Systems rebooted and database started
2. RF1 Refresh Transaction
3. Stream 00 Execution
4. RF2 Refresh Transaction

6.3. Timing Intervals for Each Query and Refresh Function

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

See Numerical Quantities Summary in the Executive Summary.

6.4. Number of Streams for the Throughput Test

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

See Numerical Quantities Summary in the Executive Summary.

6.5. Start and End Date/Times for Each Query Stream

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

See Numerical Quantities Summary in the Executive Summary.

6.6. Total Elapsed Time for the Measurement Interval

The total elapsed time of the measurement interval (see Clause 5.3.6) must be reported for the throughput test.

See Numerical Quantities Summary in the Executive Summary.

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.

See Numerical Quantities Summary in the Executive Summary

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

The timing intervals (see Clause 5.3.7) for each query of each stream and for each refresh function must be reported for the throughput test.

See Numerical Quantities Summary in the Executive Summary.

6.9. Performance Metrics

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

See Numerical Quantities Summary in the Executive Summary.

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:

	QppH@10000GB	QthH@10000GB	QphH@10000GB
Run 1	132,295.0	81,914.1	104,100.1
Run 2	133,033.8	81,813.5	104,326.2

6.11. System Activity between Tests

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

The systems were rebooted and the database server was restarted between runs.

7.0 Clause 6: SUT and Driver Implementation

7.1. Driver

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

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

The power test is invoked by calling tpcdbatch with the stream number 0 specified. The refresh function is initiated as a separate call to tpcdbatch with the SQL script for the refresh functions.

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

7.2. Implementation Specific Layer

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

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

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

For queries, each query is prepared, described, and a cursor is opened and used to fetch the required number of rows. After the last row has been retrieved a commit is issued. For the refresh functions, during the database build all data is first split for each node using the db2split utility. For RF1, the data for each node is further split into n equal portions for both the lineitem and orders tables taking care that the records for the same orderkey remain in the same set. For RF2, the data for each node is further split into m equal portions. During the run, when tpcdbatch encounters a call to execute RF1, it first calls a shell script which loads these n sets of data into staging tables. Then tpcdbatch forks off q children (where q * y = n), with each child to do y sets of insert with fullselect from the staging tables into the original lineitem and orders tables. When tpcdbatch encounters a call to execute RF2, it calls a shell script that loads these data into a single staging table. Then tpcdbatch forks off p children (where p * x =m) to do x sets of deletes from the orders and lineitem tables with a subselect from the staging table.

7.3. Profile-Directed Optimization

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

Profile-directed optimization was not used.

8.0 Clause 7: Pricing-Related Items

8.1. Hardware and Software Used

A detailed list of hardware and software used in the priced system must be reported. Each item must have a vendor part number, description, and release/revision level, and indicate General Availability (see Clause 7.2.2.1) either implicitly or explicitly(omitted Availability Dates default to the System Availability 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.

The detailed list of all hardware and software for the priced configuration is listed in the Executive Summary.

8.2. Three Year Cost of System Configuration

The total 3-year price of the entire configuration must be reported, including: hardware, software, hardware maintenance, and software support charges. Separate component pricing is required (see Clause 7.3.1. Pricing Spreadsheet.) Hardware maintenance and software support must be reported separately. The software support level must be disclosed separately from that of hardware, with separate pricing and discounts.

The price sheet for this disclosure is contained in the executive summary pages.

The pricing spreadsheet includes maintenance costs for 3 years. This service provides 7 days per week, 24 hours per day coverage.

Discounts are based on US list prices and for similar quantities and configurations. A discount of 40% has been applied to all IBM hardware, software, and services based on the total value and quantities of the components of the configuration, including full payment of all components and maintenance.

The prices listed for the IBM software products includes software support that provides the items identified in paragraph 7.1.5.6 of the TPC-H Benchmark Specification.

For assistance with any of these prices or their applicability to any customer's requirements, please contact one of the following individuals:

Bill Casey, @server pSeries Offering Manager
email: wrcasey@us.ibm.com phone 1-512-838-1422

Bernard Spang, Director, Database Market Management
email: spang@us.ibm.com phone1-914-766-1491

8.3. System Availability Date

The System Availability Date (see Clause 7.2.2.1) must be the single availability date reported on the first page of the executive summary. The full disclosure report must report Availability Dates individually for at least each of the categories for which a pricing subtotal must be provided (see Clause 7.3.1.4). All Availability Dates required to be reported must be disclosed to a precision of 1 day, but the precise format is left to the test sponsor.

The System Availability Date is August 15, 2005. All hardware (server and storage) components are available immediately. Operating system software is available immediately. Database software is available August 15, 2005.

9.0 Clause 9: Audit Items

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 Tunable Parameters

A.1 DB2 Database Configuration

Database Configuration for Database tpcd

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

Database territory = US
Database code page = 819
Database code set = ISO8859-1
Database country/region code = 1
Database collating sequence = BINARY
Alternate collating sequence (ALT_COLLATE) =
Database page size = 4096

Dynamic SQL Query management (DYN_QUERY_MGMT) = DISABLE

Discovery support for this database (DISCOVER_DB) = ENABLE

Default query optimization class (DFT_QUERYOPT) = 7
Degree of parallelism (DFT_DEGREE) = 1
Continue upon arithmetic exceptions (DFT_SQLMATHWARN) = NO
Default refresh age (DFT_REFRESH_AGE) = 0
Default maintained table types for opt (DFT_MTTB_TYPES) = SYSTEM
Number of frequent values retained (NUM_FREQVALUES) = 0
Number of quantiles retained (NUM_QUANTILES) = 600

Backup pending = NO

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

Multi-page file allocation enabled = YES

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

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

Database heap (4KB) (DBHEAP) = 1500
Size of database shared memory (4KB) (DATABASE_MEMORY) = 1030000
Catalog cache size (4KB) (CATALOGCACHE_SZ) = (MAXAPPLS*4)
Log buffer size (4KB) (LOGBUFSZ) = 1024
Utilities heap size (4KB) (UTIL_HEAP_SZ) = 5000
Buffer pool size (pages) (BUFFPAGE) = 1000
Extended storage segments size (4KB) (ESTORE_SEG_SZ) = 16000
Number of extended storage segments (NUM_ESTORE_SEGS) = 0
Max storage for lock list (4KB) (LOCKLIST) = 32768

Max size of appl. group mem set (4KB) (APPGROUP_MEM_SZ) = 40000
Percent of mem for appl. group heap (GROUPHEAP_RATIO) = 70
Max appl. control heap size (4KB) (APP_CTL_HEAP_SZ) = 512

Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = 250
Sort list heap (4KB) (SORTHEAP) = 50000
SQL statement heap (4KB) (STMTHEAP) = 20000
Default application heap (4KB) (APPLHEAPSZ) = 156
Package cache size (4KB) (PCKCACHESZ) = (MAXAPPLS*8)
Statistics heap size (4KB) (STAT_HEAP_SZ) = 4384

Interval for checking deadlock (ms) (DLCHKTIME) = 60000
Percent. of lock lists per application (MAXLOCKS) = 13
Lock timeout (sec) (LOCKTIMEOUT) = -1

Changed pages threshold (CHNGPGS_THRESH) = 70
Number of asynchronous page cleaners (NUM_IOCLEANERS) = 2
Number of I/O servers (NUM_IOSERVERS) = 3
Index sort flag (INDEXSORT) = YES
Sequential detect flag (SEQDETECT) = YES
Default prefetch size (pages) (DFT_PREFETCH_SZ) = AUTOMATIC

Track modified pages (TRACKMOD) = OFF

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

Max number of active applications (MAXAPPLS) = 365
Average number of active applications (AVG_APPLS) = 1
Max DB files open per application (MAXFILOP) = 1024

Log file size (4KB) (LOGFILSIZ) = 8192
Number of primary log files (LOGPRIMARY) = 120
Number of secondary log files (LOGSECOND) = 5
Changed path to log files (NEWLOGPATH) =
Path to log files = /db2/tpcd/NODE0001/
Overflow log path (OVERFLOWLOGPATH) =
Mirror log path (MIRRORLOGPATH) =
First active log file =
Block log on disk full (BLK_LOG_DSK_FUL) = NO
Percent of max active log space by transaction(MAX_LOG) = 0
Num. of active log files for 1 active UOW(NUM_LOG_SPAN) = 0

Group commit count (MINCOMMIT) = 1
Percent log file reclaimed before soft chckpt (SOFTMAX) = 300
Log retain for recovery enabled (LOGRETAIN) = OFF
User exit for logging enabled (USEREXIT) = OFF

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

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

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

```
Recovery history retention (days)     (REC_HIS_RETENTN) = 366

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

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

A.2 DB2 Database Manager Configuration

```
   Database Manager Configuration

   Node type = Enterprise Server Edition with local and remote clients

Database manager configuration release level            = 0x0a00

CPU speed (millisec/instruction)             (CPUSPEED) = 1.000000e-08
Communications bandwidth (MB/sec)      (COMM_BANDWIDTH) =
2.000000e+02

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

Default charge-back account          (DFT_ACCOUNT_STR) =

Java Development Kit installation path       (JDK_PATH) =
/usr/java14_64

Diagnostic error capture level             (DIAGLEVEL) = 3
Notify Level                          (NOTIFYLEVEL) = 3
Diagnostic data directory path              (DIAGPATH) =
/db2dump/db2dump

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

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

Client Userid-Password Plugin          (CLNT_PW_PLUGIN) =
Client Kerberos Plugin                 (CLNT_KRB_PLUGIN) =
Group Plugin                           (GROUP_PLUGIN) =
GSS Plugin for Local Authorization     (LOCAL_GSSPLUGIN) =
Server Plugin Mode                     (SRV_PLUGIN_MODE) =
UNFENCED
Server List of GSS Plugins       (SRVCON_GSSPLUGIN_LIST) =
Server Userid-Password Plugin          (SRVCON_PW_PLUGIN) =
Server Connection Authentication           (SRVCON_AUTH) =
NOT_SPECIFIED

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

Default database path                      (DFTDBPATH) = /tpc/tpcd

Database monitor heap size (4KB)           (MON_HEAP_SZ) = 90
Java Virtual Machine heap size (4KB)       (JAVA_HEAP_SZ) = 1024
Audit buffer size (4KB)                    (AUDIT_BUF_SZ) = 0
Size of instance shared memory (4KB)  (INSTANCE_MEMORY) =
24576
Backup buffer default size (4KB)           (BACKBUFSZ) = 1024
Restore buffer default size (4KB)          (RESTBUFSZ) = 1024

Sort heap threshold (4KB)                  (SHEAPTHRES) = 1400000

Directory cache support                    (DIR_CACHE) = YES

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

Workload impact by throttled utilities(UTIL_IMPACT_LIM) = 10

Priority of agents                         (AGENTPRI) = SYSTEM
Max number of existing agents              (MAXAGENTS) = 400
Agent pool size                       (NUM_POOLAGENTS) = 0
Initial number of agents in pool       (NUM_INITAGENTS) = 0
Max number of coordinating agents     (MAX_COORDAGENTS) =
(MAXAGENTS - NUM_INITAGENTS)
Max no. of concurrent coordinating agents  (MAXCAGENTS) =
MAX_COORDAGENTS
Max number of client connections      (MAX_CONNECTIONS) =
MAX_COORDAGENTS

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

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

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

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

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

Maximum query degree of parallelism   (MAX_QUERYDEGREE) =
ANY
Enable intra-partition parallelism     (INTRA_PARALLEL) = NO

No. of int. communication buffers(4KB)(FCM_NUM_BUFFERS) =
147456
Number of FCM request blocks               (FCM_NUM_RQB) = 16384
Number of FCM connection entries      (FCM_NUM_CONNECT) =
16384
Number of FCM message anchors         (FCM_NUM_ANCHORS) =
4096
```

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

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

A.3 DB2 Registry Settings

```
DB2_LARGE_PAGE_MEM=DBMS,FCM,DB
DB2_EVENT_LOG_CONFIG=OFF
DB2_EXTENDED_OPTIMIZATION=Y
DB2_ANTIJOIN=ON
DB2BPVARS=/tpc/tpcd/tpcd/custom/bpvars.cfg
DB2_FORCE_FCM_BP=ON
DB2OPTIONS=-t -v +c
DB2_PARALLEL_IO=*

/tpc/tpcd/tpcd/custom/bpvars.cfg Content
NUMPREFETCHQUEUES=1
PREFETCHQUEUESIZE=200
```

A.4 AIX Parameters

```
vmo -r -o minperm%=15
vmo -r -o maxperm%=25
vmo -r -o maxclient%=15
vmo -r -o maxfree=1088
vmo -r -o v_pinshm=1
vmo -r -o lgpg_size=16M
vmo -r -o lgpg_regions=4238

ioo -r -o lvm_bufcnt=16
ioo -r -o maxpgahead=64

chdev -l sys0 -a maxuproc=8000

chuser capabilities=CAP_BYPASS_RAC_VMM,CAP_PROPAGATE
,CAP_NUMA_ATTACH tpcd

node1:
DB2/MLN1:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00000
DB2/MLN2:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00001
DB2/MLN3:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00002
DB2/MLN4:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00003
DB2/MLN5:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00004
DB2/MLN6:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00005
DB2/MLN7:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00006
DB2/MLN8:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00007
DB2/MLN9:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00000
DB2/MLN10:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00001
DB2/MLN11:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00002
DB2/MLN12:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00003
DB2/MLN13:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00004
DB2/MLN14:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00005
DB2/MLN15:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00006
DB2/MLN16:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00007

node2:
DB2/MLN17:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00000
DB2/MLN18:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00001
DB2/MLN19:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00002
DB2/MLN20:
   owner     = tpcd
```

```
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00003
DB2/MLN21:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00004
DB2/MLN22:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00005
DB2/MLN23:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00006
DB2/MLN24:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00007
DB2/MLN25:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00000
DB2/MLN26:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00001
DB2/MLN27:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00002
DB2/MLN28:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00003
DB2/MLN29:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00004
DB2/MLN30:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00005
DB2/MLN31:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00006
DB2/MLN32:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00007

node3:
DB2/MLN33:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00000
DB2/MLN34:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00001
DB2/MLN35:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00002
DB2/MLN36:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00003
DB2/MLN37:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00004
DB2/MLN38:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00005
DB2/MLN39:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00006
DB2/MLN40:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00007
DB2/MLN41:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00000
DB2/MLN42:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00001
DB2/MLN43:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00002
DB2/MLN44:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00003
DB2/MLN45:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00004
DB2/MLN46:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00005
DB2/MLN47:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00006
```

```
DB2/MLN48:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00007

node4:
DB2/MLN49:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00000
DB2/MLN50:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00001
DB2/MLN51:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00002
DB2/MLN52:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00003
DB2/MLN53:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00004
DB2/MLN54:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00005
DB2/MLN55:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00006
DB2/MLN56:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00007
DB2/MLN57:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00000
DB2/MLN58:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00001
DB2/MLN59:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00002
DB2/MLN60:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00003
DB2/MLN61:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00004
DB2/MLN62:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00005
DB2/MLN63:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00006
DB2/MLN64:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00007

node5:
DB2/MLN65:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00000
DB2/MLN66:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00001
DB2/MLN67:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00002
DB2/MLN68:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00003
DB2/MLN69:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00004
DB2/MLN70:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00005
DB2/MLN71:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00006
DB2/MLN72:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00007
DB2/MLN73:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00000
DB2/MLN74:
   owner     = tpcd
   group     = system
   perm      = rwr-r-
   resources = sys/cpu.00001
DB2/MLN75:
```

```
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00002
DB2/MLN76:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00003
DB2/MLN77:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00004
DB2/MLN78:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00005
DB2/MLN79:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00006
DB2/MLN80:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00007

node6:
DB2/MLN81:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00000
DB2/MLN82:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00001
DB2/MLN83:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00002
DB2/MLN84:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00003
DB2/MLN85:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00004
DB2/MLN86:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00005
DB2/MLN87:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00006
DB2/MLN88:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00007
DB2/MLN89:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00000
DB2/MLN90:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00001
DB2/MLN91:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00002
DB2/MLN92:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00003
DB2/MLN93:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00004
DB2/MLN94:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00005
DB2/MLN95:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00006
DB2/MLN96:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00007

node7:
DB2/MLN97:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00000
DB2/MLN98:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00001
DB2/MLN99:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00002
DB2/MLN100:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00003
DB2/MLN101:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00004
DB2/MLN102:
    owner     = tpcd
```

```
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00005
DB2/MLN103:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00006
DB2/MLN104:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00007
DB2/MLN105:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00000
DB2/MLN106:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00001
DB2/MLN107:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00002
DB2/MLN108:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00003
DB2/MLN109:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00004
DB2/MLN110:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00005
DB2/MLN111:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00006
DB2/MLN112:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00007

node8:
DB2/MLN113:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00000
DB2/MLN114:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00001
DB2/MLN115:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00002
DB2/MLN116:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00003
DB2/MLN117:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00004
DB2/MLN118:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00005
DB2/MLN119:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00006
DB2/MLN120:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00007
DB2/MLN121:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00000
DB2/MLN122:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00001
DB2/MLN123:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00002
DB2/MLN124:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00003
DB2/MLN125:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00004
DB2/MLN126:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00005
DB2/MLN127:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00006
DB2/MLN128:
    owner     = tpcd
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00007
```

Appendix - B Database Build Scripts

B.1 db2nodes.cfg

```
1   sagitta1a   0   sagitta1a DB2/MLN1
2   sagitta1a   1   sagitta1a DB2/MLN2
3   sagitta1a   2   sagitta1a DB2/MLN3
4   sagitta1a   3   sagitta1a DB2/MLN4
5   sagitta1a   4   sagitta1a DB2/MLN5
6   sagitta1a   5   sagitta1a DB2/MLN6
7   sagitta1a   6   sagitta1a DB2/MLN7
8   sagitta1a   7   sagitta1a DB2/MLN8
9   sagitta1a   8   sagitta1b DB2/MLN9
10  sagitta1a   9   sagitta1b DB2/MLN10
11  sagitta1a   10  sagitta1b DB2/MLN11
12  sagitta1a   11  sagitta1b DB2/MLN12
13  sagitta1a   12  sagitta1b DB2/MLN13
14  sagitta1a   13  sagitta1b DB2/MLN14
15  sagitta1a   14  sagitta1b DB2/MLN15
16  sagitta1a   15  sagitta1b DB2/MLN16
17  sagitta2a   0   sagitta2a DB2/MLN17
18  sagitta2a   1   sagitta2a DB2/MLN18
19  sagitta2a   2   sagitta2a DB2/MLN19
20  sagitta2a   3   sagitta2a DB2/MLN20
21  sagitta2a   4   sagitta2a DB2/MLN21
22  sagitta2a   5   sagitta2a DB2/MLN22
23  sagitta2a   6   sagitta2a DB2/MLN23
24  sagitta2a   7   sagitta2a DB2/MLN24
25  sagitta2a   8   sagitta2b DB2/MLN25
26  sagitta2a   9   sagitta2b DB2/MLN26
27  sagitta2a   10  sagitta2b DB2/MLN27
28  sagitta2a   11  sagitta2b DB2/MLN28
29  sagitta2a   12  sagitta2b DB2/MLN29
30  sagitta2a   13  sagitta2b DB2/MLN30
31  sagitta2a   14  sagitta2b DB2/MLN31
32  sagitta2a   15  sagitta2b DB2/MLN32
33  sagitta3a   0   sagitta3a DB2/MLN33
34  sagitta3a   1   sagitta3a DB2/MLN34
35  sagitta3a   2   sagitta3a DB2/MLN35
36  sagitta3a   3   sagitta3a DB2/MLN36
37  sagitta3a   4   sagitta3a DB2/MLN37
38  sagitta3a   5   sagitta3a DB2/MLN38
39  sagitta3a   6   sagitta3a DB2/MLN39
40  sagitta3a   7   sagitta3a DB2/MLN40
41  sagitta3a   8   sagitta3b DB2/MLN41
42  sagitta3a   9   sagitta3b DB2/MLN42
43  sagitta3a   10  sagitta3b DB2/MLN43
44  sagitta3a   11  sagitta3b DB2/MLN44
45  sagitta3a   12  sagitta3b DB2/MLN45
46  sagitta3a   13  sagitta3b DB2/MLN46
47  sagitta3a   14  sagitta3b DB2/MLN47
48  sagitta3a   15  sagitta3b DB2/MLN48
49  sagitta4a   0   sagitta4a DB2/MLN49
50  sagitta4a   1   sagitta4a DB2/MLN50
51  sagitta4a   2   sagitta4a DB2/MLN51
52  sagitta4a   3   sagitta4a DB2/MLN52
53  sagitta4a   4   sagitta4a DB2/MLN53
54  sagitta4a   5   sagitta4a DB2/MLN54
55  sagitta4a   6   sagitta4a DB2/MLN55
56  sagitta4a   7   sagitta4a DB2/MLN56
57  sagitta4a   8   sagitta4b DB2/MLN57
58  sagitta4a   9   sagitta4b DB2/MLN58
59  sagitta4a   10  sagitta4b DB2/MLN59
60  sagitta4a   11  sagitta4b DB2/MLN60
61  sagitta4a   12  sagitta4b DB2/MLN61
62  sagitta4a   13  sagitta4b DB2/MLN62
63  sagitta4a   14  sagitta4b DB2/MLN63
64  sagitta4a   15  sagitta4b DB2/MLN64
65  sagitta5a   0   sagitta5a DB2/MLN65
66  sagitta5a   1   sagitta5a DB2/MLN66
67  sagitta5a   2   sagitta5a DB2/MLN67
68  sagitta5a   3   sagitta5a DB2/MLN68
69  sagitta5a   4   sagitta5a DB2/MLN69
70  sagitta5a   5   sagitta5a DB2/MLN70
71  sagitta5a   6   sagitta5a DB2/MLN71
72  sagitta5a   7   sagitta5a DB2/MLN72
73  sagitta5a   8   sagitta5b DB2/MLN73
74  sagitta5a   9   sagitta5b DB2/MLN74
75  sagitta5a   10  sagitta5b DB2/MLN75
76  sagitta5a   11  sagitta5b DB2/MLN76
77  sagitta5a   12  sagitta5b DB2/MLN77
78  sagitta5a   13  sagitta5b DB2/MLN78
79  sagitta5a   14  sagitta5b DB2/MLN79
80  sagitta5a   15  sagitta5b DB2/MLN80
81  sagitta6a   0   sagitta6a DB2/MLN81
82  sagitta6a   1   sagitta6a DB2/MLN82
83  sagitta6a   2   sagitta6a DB2/MLN83
84  sagitta6a   3   sagitta6a DB2/MLN84
85  sagitta6a   4   sagitta6a DB2/MLN85
86  sagitta6a   5   sagitta6a DB2/MLN86
87  sagitta6a   6   sagitta6a DB2/MLN87
88  sagitta6a   7   sagitta6a DB2/MLN88
89  sagitta6a   8   sagitta6b DB2/MLN89
90  sagitta6a   9   sagitta6b DB2/MLN90
91  sagitta6a   10  sagitta6b DB2/MLN91
92  sagitta6a   11  sagitta6b DB2/MLN92
93  sagitta6a   12  sagitta6b DB2/MLN93
94  sagitta6a   13  sagitta6b DB2/MLN94
95  sagitta6a   14  sagitta6b DB2/MLN95
96  sagitta6a   15  sagitta6b DB2/MLN96
97  sagitta7a   0   sagitta7a DB2/MLN97
98  sagitta7a   1   sagitta7a DB2/MLN98
99  sagitta7a   2   sagitta7a DB2/MLN99
100 sagitta7a   3   sagitta7a DB2/MLN100
101 sagitta7a   4   sagitta7a DB2/MLN101
102 sagitta7a   5   sagitta7a DB2/MLN102
103 sagitta7a   6   sagitta7a DB2/MLN103
104 sagitta7a   7   sagitta7a DB2/MLN104
105 sagitta7a   8   sagitta7b DB2/MLN105
106 sagitta7a   9   sagitta7b DB2/MLN106
107 sagitta7a   10  sagitta7b DB2/MLN107
108 sagitta7a   11  sagitta7b DB2/MLN108
109 sagitta7a   12  sagitta7b DB2/MLN109
110 sagitta7a   13  sagitta7b DB2/MLN110
111 sagitta7a   14  sagitta7b DB2/MLN111
112 sagitta7a   15  sagitta7b DB2/MLN112
113 sagitta8a   0   sagitta8a DB2/MLN113
114 sagitta8a   1   sagitta8a DB2/MLN114
115 sagitta8a   2   sagitta8a DB2/MLN115
116 sagitta8a   3   sagitta8a DB2/MLN116
117 sagitta8a   4   sagitta8a DB2/MLN117
118 sagitta8a   5   sagitta8a DB2/MLN118
119 sagitta8a   6   sagitta8a DB2/MLN119
120 sagitta8a   7   sagitta8a DB2/MLN120
121 sagitta8a   8   sagitta8b DB2/MLN121
122 sagitta8a   9   sagitta8b DB2/MLN122
123 sagitta8a   10  sagitta8b DB2/MLN123
124 sagitta8a   11  sagitta8b DB2/MLN124
125 sagitta8a   12  sagitta8b DB2/MLN125
126 sagitta8a   13  sagitta8b DB2/MLN126
127 sagitta8a   14  sagitta8b DB2/MLN127
128 sagitta8a   15  sagitta8b DB2/MLN128
```

B.2 db2set_load.sh

```
db2set DB2_LARGE_PAGE_MEM=DBMS,FCM,DB
db2set DB2_EVENT_LOG_CONFIG=OFF
db2set DB2_EXTENDED_OPTIMIZATION=Y
db2set DB2_ANTIJOIN=ON
db2set DB2BPVARS=/tpc/tpcd/tpcd/custom/bpvars.cfg
db2set DB2_FORCE_FCM_BP=ON
db2set DB2OPTIONS='-t -v +c'
db2set DB2_PARALLEL_IO=*
```

B.3 /tpc/tpcd/tpcd/custom/bpvar.cfg

```
NUMPREFETCHQUEUES=1
PREFETCHQUEUESIZE=200
```

B.4 dbmcfg_load.clp

```
update database manager configuration using
numdb                1
diagpath               /db2dump/db2dump
cpuspeed           1e-8
comm_bandwidth       200
sheapthres       1400000
instance_memory    24576
max_querydegree       -1
health_mon             off
fcm_num_buffers   147456
fcm_num_rqb        16384
fcm_num_connect    16384
fcm_num_anchors     4096
num_initagents         0
num_poolagents         0
intra_parallel          no;
```

B.5 dbcfg_load.clp

```
update database configuration for tpcd using
dft_queryopt 7
num_freqvalues 0
num_quantiles 600
stat_heap_sz 30000
sheapthres_shr 250
dbheap 1500
util_heap_sz 65536
database_memory 1030000
locklist 4096
app_ctl_heap_sz 512
sortheap 50000
stmtheap 20000
applheapsz 156
maxlocks 13
dlchktime 60000
chngpgs_thresh 70
num_iocleaners 2
num_ioservers 3
maxappls 365
maxfilop 1024
newlogpath /db2/tpcd
logbufsz 1024
softmax 300
logfilsiz 8192
logprimary 120
logsecond 5;
```

B.6 ng.ddl

```
CREATE NODEGROUP ng_all ON NODES (1 to 128);
COMMIT WORK;
CREATE NODEGROUP ng_node1 ON NODE (1);
COMMIT WORK;
```

B.7 bp.ddl

```
ALTER BUFFERPOOL IBMDEFAULTBP DEFERRED SIZE 143000;
COMMIT WORK;
CREATE BUFFERPOOL BP32KTMP DEFERRED ALL NODES SIZE
18000 PAGESIZE 32K;
COMMIT WORK;
CREATE BUFFERPOOL BP32K DEFERRED ALL NODES SIZE
84900 PAGESIZE 32K;
COMMIT WORK;
```

B.8 tbsp.ddl

```
CREATE regular TABLESPACE TABDATA
  in nodegroup ng_all
  PAGESIZE 32K MANAGED BY DATABASE
USING (
device '/dev/rdata1p $N' 1067008,
device '/dev/rdata2p $N' 1067008,
device '/dev/rdata3p $N' 1067008
)
BUFFERPOOL BP32K
extentsize 16
prefetchsize 144
overhead 6.0
transferrate 0.29 ;
commit;

CREATE regular TABLESPACE TABINDEXES
  in nodegroup ng_all
  PAGESIZE 32K MANAGED BY DATABASE
USING (
device '/dev/ridx1p $N' 206848,
device '/dev/ridx2p $N' 206848,
device '/dev/ridx3p $N' 206848
)
BUFFERPOOL BP32K
extentsize 16
prefetchsize 144
overhead 6.0
transferrate 0.29;
commit;

CREATE TEMPORARY TABLESPACE TEMPSP4K
  PAGESIZE 4K MANAGED BY DATABASE
USING (
device '/dev/rjbtemp1p $N'  6094848,
device '/dev/rjbtemp2p $N'  6094848,
device '/dev/rjbtemp3p $N'  6094848
)
BUFFERPOOL IBMDEFAULTBP
extentsize 16
prefetchsize 144
overhead 6.0
transferrate 0.29;
commit;

CREATE TEMPORARY TABLESPACE TEMPSP32K
  PAGESIZE 32K MANAGED BY DATABASE
USING (
device '/dev/rjbtm1p $N'  327680,
device '/dev/rjbtm2p $N'  327680,
```

```
device '/dev/rjbtm3p $N'  327680
)
BUFFERPOOL BP32KTMP
extentsize 16
prefetchsize 144
overhead 6.0
transferrate 0.29;
commit;

CREATE regular TABLESPACE TABSTAGE
  in nodegroup ng_all
  PAGESIZE 32K MANAGED BY DATABASE
USING (
device '/dev/rstg1p $N' 4096,
device '/dev/rstg2p $N' 4096,
device '/dev/rstg3p $N' 4096
)
BUFFERPOOL BP32KTMP
extentsize 16
prefetchsize 144
overhead 6.0
transferrate 0.29;
commit;

CREATE REGULAR TABLESPACE SMALL
  in nodegroup ng_node1
  pagesize 4K
  managed by system
  using ('/db/small')
  bufferpool IBMDEFAULTBP;
COMMIT WORK;

drop TABLESPACE TEMPSPACE1;
commit;
```

B.9 UFtbl.ddl

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

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

CREATE TABLE TPCDTEMP.LINEITEM_NEW ( APP_ID INTEGER
NOT NULL,
                 L_ORDERKEY    BIGINT NOT NULL,
                 L_PARTKEY     INTEGER NOT NULL,
                 L_SUPPKEY     INTEGER NOT NULL,
                 L_LINENUMBER  INTEGER NOT NULL,
                 L_QUANTITY    FLOAT NOT NULL,
                 L_EXTENDEDPRICE  FLOAT NOT NULL,
                 L_DISCOUNT    FLOAT NOT NULL,
                 L_TAX         FLOAT NOT NULL,
                 L_RETURNFLAG  CHAR(1) NOT NULL,
                 L_LINESTATUS  CHAR(1) NOT NULL,
                 L_SHIPDATE    DATE NOT NULL,
                 L_COMMITDATE  DATE NOT NULL,
                 L_RECEIPTDATE DATE NOT NULL,
                 L_SHIPINSTRUCT CHAR(25) NOT NULL,
                 L_SHIPMODE     CHAR(10) NOT NULL,
                 L_COMMENT      VARCHAR(44) NOT NULL WITH
DEFAULT)
     PARTITIONING KEY (L_ORDERKEY)
     IN TABSTAGE ;

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)
PCTFREE 0;

CREATE INDEX TPCDTEMP.I_LINEITEM_NEW ON
TPCDTEMP.LINEITEM_NEW (APP_ID) PCTFREE 0;

CREATE UNIQUE INDEX TPCDTEMP.I_ORDERS_DEL ON
TPCDTEMP.ORDERS_DEL (APP_ID, O_ORDERKEY) PCTFREE 0;
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;
```

B.10 UFtbl_load.sh

```
#!/bin/ksh
toolsDir=/tpc/tpcd/tpcd/tools
${toolsDir}/ploaduf1 32
${toolsDir}/ploaduf2 32
db2 connect to tpcd
db2 RUNSTATS ON TABLE TPCDTEMP.LINEITEM_NEW WITH
DISTRIBUTION AND DETAILED INDEXES ALL
db2 RUNSTATS ON TABLE TPCDTEMP.ORDERS_NEW WITH
DISTRIBUTION AND DETAILED INDEXES ALL
db2 RUNSTATS ON TABLE TPCDTEMP.ORDERS_DEL WITH
DISTRIBUTION AND DETAILED INDEXES ALL
db2 connect reset
db2 terminate
```

B.11 ploaduf1

```
#!/bin/ksh
toolsDir=/tpc/tpcd/tpcd/tools
InDir=/rawdata1/LINKS/uflinks
RFpair=$1;
$toolsDir/ploaduf1.gruntO $RFpair $InDir&
$toolsDir/ploaduf1.gruntL $RFpair $InDir
wait
```

B.12 ploaduf1.gruntO

```
#!/bin/ksh
RFpair=$1;
InDir=$2
```

```
db2 -c +p -svt << EOF
connect to tpcd;
load from orders.u$RFpair of del modified by coldel| fastparse
messages /dev/null replace into TPCDTEMP.ORDERS_new
nonrecoverable partitioned db config mode load_only part_file_location
$InDir;
connect reset;
terminate;
EOF
```

B.13 ploaduf1.gruntL

```
#!/bin/ksh
RFpair=$1;
InDir=$2
db2 -c +p -svt << EOF
connect to tpcd;
load from lineitem.u$RFpair of del modified by coldel| fastparse
messages /dev/null replace into TPCDTEMP.LINEITEM_new
nonrecoverable partitioned db config mode load_only part_file_location
$InDir;
connect reset;
terminate;
EOF
```

B.14 ploaduf2

```
#!/bin/ksh
RFpair=$1;
InDir=/rawdata1/LINKS/uflinks
db2 -c +p -svt << EOF
connect to tpcd;
load from delete.$RFpair of del modified by coldel| fastparse messages
/dev/null replace into TPCDTEMP.ORDERS_del nonrecoverable
partitioned db config mode load_only part_file_location $InDir;
connect reset;
terminate;
EOF
```

B.15 UFtbl_delete.ddl

```
import from /dev/null of del replace into TPCDTEMP.ORDERS_NEW;
import from /dev/null of del replace into TPCDTEMP.LINEITEM_NEW;
import from /dev/null of del replace into TPCDTEMP.ORDERS_DEL;
```

B.16 tbl.ddl

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

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

CREATE TABLE TPCD.PART  ( P_PARTKEY     INTEGER NOT
NULL,
                P_NAME       VARCHAR(55) NOT NULL,
                P_MFGR       CHAR(25) NOT NULL,
                P_BRAND      CHAR(10) NOT NULL,
                P_TYPE       VARCHAR(25) NOT NULL,
                P_SIZE       INTEGER NOT NULL,
                P_CONTAINER  CHAR(10) NOT NULL,
                P_RETAILPRICE FLOAT NOT NULL,
                P_COMMENT    VARCHAR(23) NOT NULL )
    IN TABDATA
    INDEX IN TABINDEXES;

CREATE TABLE TPCD.SUPPLIER ( S_SUPPKEY     INTEGER NOT
NULL,
                  S_NAME       CHAR(25) NOT NULL,
                  S_ADDRESS    VARCHAR(40) NOT NULL,
                  S_NATIONKEY  INTEGER NOT NULL,
                  S_PHONE      CHAR(15) NOT NULL,
                  S_ACCTBAL    FLOAT NOT NULL,
                  S_COMMENT    VARCHAR(101) NOT NULL)
    IN TABDATA
    INDEX IN TABINDEXES;

CREATE TABLE TPCD.PARTSUPP ( PS_PARTKEY     INTEGER
NOT NULL,
                  PS_SUPPKEY    INTEGER NOT NULL,
                  PS_AVAILQTY   INTEGER NOT NULL,
                  PS_SUPPLYCOST  FLOAT   NOT NULL,
                  PS_COMMENT    VARCHAR(199) NOT NULL )
    IN TABDATA
    INDEX IN TABINDEXES;

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

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)
ORGANIZE BY (O_ORDERDATE)
    IN TABDATA
    INDEX IN TABINDEXES;

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)
```

```
ORGANIZE BY (L_SHIPDATE)
    IN TABDATA
    INDEX IN TABINDEXES;

COMMIT WORK;
```

B.17 load.sh

```
#!/bin/ksh
messages=${TPCD_TMP_DIR}
rawdata=${TPCD_INPUT}
custom=${TPCD_DDLPATH}

echo "Load Summary Time: " > ${messages}/loadstatus.out

db2 connect to tpcd;

echo "Loading Nation at "`date` >> ${messages}/loadstatus.out
db2 "load from ${rawdata}1/nation.tbl of del modified by coldel|
fastparse noheader messages /dev/null replace i
nto TPCD.NATION nonrecoverable cpu_parallelism 1"

echo "Loading Region at "`date` >> ${messages}/loadstatus.out
db2 "load from ${rawdata}1/region.tbl of del modified by coldel|
fastparse noheader messages /dev/null replace i
nto TPCD.REGION nonrecoverable cpu_parallelism 1"

echo "Loading Lineitem at "`date` >> ${messages}/loadstatus.out
 db2 -tvf ${custom}/load_lineitem.ddl
#
echo "Loading Orders at "`date` >> ${messages}/loadstatus.out
 db2 -tvf ${custom}/load_orders.ddl
#
echo "Loading Partsupp at "`date` >> ${messages}/loadstatus.out
 db2 -tvf ${custom}/load_partsupp.ddl
#
echo "Loading Customer at "`date` >> ${messages}/loadstatus.out
 db2 -tvf ${custom}/load_customer.ddl
#
echo "Loading Part at "`date` >> ${messages}/loadstatus.out
 db2 -tvf ${custom}/load_part.ddl
#
echo "Loading Supplier at "`date` >> ${messages}/loadstatus.out
 db2 -tvf ${custom}/load_supplier.ddl
#

db2 commit;
db2 terminate;
echo "Finished Loading  at "`date` >> ${messages}/loadstatus.out
echo "------------------------------------------" >> ${messages}/loadstatus.out
```

B.18 load_lineitem.clp

```
load from
   lineitem.1,   lineitem.2,   lineitem.3,   lineitem.4,
   lineitem.5,   lineitem.6,   lineitem.7,   lineitem.8,
   lineitem.9,   lineitem.10,  lineitem.11,  lineitem.12,
   lineitem.13,  lineitem.14,  lineitem.15,  lineitem.16,
   lineitem.17,  lineitem.18,  lineitem.19,  lineitem.20,
   lineitem.21,  lineitem.22,  lineitem.23,  lineitem.24,
   lineitem.25,  lineitem.26,  lineitem.27,  lineitem.28,
   lineitem.29,  lineitem.30,   lineitem.31, lineitem.32
   of del modified by coldel|
    fastparse
    messages /dev/null
    replace into TPCD.lineitem nonrecoverable cpu_parallelism 1
    partitioned db config mode load_only
    part_file_location /rawdata1/LINKS/load_links;
```

B.19 load_orders.clp

```
load from
   orders.1,   orders.2,  orders.3,  orders.4,
   orders.5,   orders.6,  orders.7,  orders.8,
   orders.9,   orders.10, orders.11, orders.12,
   orders.13, orders.14, orders.15, orders.16,
   orders.17,  orders.18, orders.19, orders.20,
   orders.21,  orders.22, orders.23, orders.24,
   orders.25,  orders.26, orders.27, orders.28,
   orders.29,  orders.30, orders.31, orders.32
   of del modified by coldel|
    fastparse
    messages /dev/null
    replace into TPCD.orders nonrecoverable cpu_parallelism 1
    partitioned db config mode load_only
    part_file_location /rawdata1/LINKS/load_links;
```

B.20 load_partsupp.clp

```
load from
   partsupp.1, partsupp.2, partsupp.3, partsupp.4,
   partsupp.5, partsupp.6, partsupp.7, partsupp.8,
   partsupp.9, partsupp.10, partsupp.11, partsupp.12,
   partsupp.13, partsupp.14, partsupp.15, partsupp.16,
   partsupp.17, partsupp.18, partsupp.19, partsupp.20,
   partsupp.21, partsupp.22, partsupp.23, partsupp.24,
   partsupp.25, partsupp.26, partsupp.27, partsupp.28,
   partsupp.29, partsupp.30, partsupp.31, partsupp.32
   of del modified by coldel|
    fastparse
    messages /dev/null
    replace into TPCD.partsupp nonrecoverable cpu_parallelism 1
    partitioned db config mode load_only
    part_file_location /rawdata1/LINKS/load_links;
```

B.21 load_part.clp

```
load from
   part.1, part.2, part.3, part.4,
   part.5, part.6, part.7, part.8,
   part.9, part.10, part.11, part.12,
   part.13, part.14, part.15, part.16,
   part.17, part.18, part.19, part.20,
   part.21, part.22, part.23, part.24,
   part.25, part.26, part.27, part.28,
   part.29, part.30, part.31, part.32
   of del modified by coldel|
    fastparse
    messages /dev/null
    replace into TPCD.part nonrecoverable cpu_parallelism 1
    partitioned db config mode load_only
    part_file_location /rawdata1/LINKS/load_links;
```

B.22 load_customer.clp

```
load from
   customer.1, customer.2, customer.3, customer.4,
   customer.5, customer.6, customer.7, customer.8,
   customer.9, customer.10, customer.11, customer.12,
   customer.13, customer.14, customer.15, customer.16,
   customer.17, customer.18, customer.19, customer.20,
   customer.21, customer.22, customer.23, customer.24,
   customer.25, customer.26, customer.27, customer.28,
   customer.29, customer.30, customer.31, customer.32
   of del modified by coldel|
    fastparse
    messages /dev/null
    replace into TPCD.customer nonrecoverable cpu_parallelism 1
```

```
    partitioned db config mode load_only
    part_file_location /rawdata1/LINKS/load_links;
```

B.23 load_supplier.clp

```
load from
   supplier.1, supplier.2, supplier.3, supplier.4,
   supplier.5, supplier.6, supplier.7, supplier.8,
   supplier.9, supplier.10, supplier.11, supplier.12,
   supplier.13, supplier.14, supplier.15, supplier.16,
   supplier.17, supplier.18, supplier.19, supplier.20,
   supplier.21, supplier.22, supplier.23, supplier.24,
   supplier.25, supplier.26, supplier.27, supplier.28,
   supplier.29, supplier.30, supplier.31, supplier.32
   of del modified by coldel|
    fastparse
    messages /dev/null
    replace into TPCD.supplier nonrecoverable cpu_parallelism 1
    partitioned db config mode load_only
    part_file_location /rawdata1/LINKS/load_links;
```

B.24 idx.ddl

```
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;

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;

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

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;

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

CREATE UNIQUE INDEX TPCD.PS_PKSK ON TPCD.PARTSUPP
            (PS_PARTKEY ASC, PS_SUPPKEY ASC) PCTFREE 0 ;
commit work;
alter table tpcd.partsupp add primary key (ps_partkey, ps_suppkey);
commit work;

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

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

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

CREATE UNIQUE INDEX TPCD.P_PK ON TPCD.PART
            (P_PARTKEY ASC) PCTFREE 0 ;
commit work;
alter table tpcd.part add primary key (p_partkey);
commit work;

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

CREATE UNIQUE INDEX TPCD.O_OK ON TPCD.ORDERS
            (O_ORDERKEY ASC) PCTFREE 3 ;
commit work;
alter table tpcd.orders add primary key (o_orderkey);
commit work;

CREATE INDEX TPCD.O_CK ON TPCD.ORDERS
            (O_CUSTKEY ASC) PCTFREE 3 ;
commit work;

CREATE UNIQUE INDEX TPCD.L_OKLN ON TPCD.LINEITEM
            (L_ORDERKEY ASC, L_LINENUMBER ASC) PCTFREE 3 ;
commit work;
alter table tpcd.lineitem add primary key (l_orderkey, l_linenumber);
commit work;
```

B.25 runstats.sh

```
#!/bin/ksh
# do half the partitioned tables in parallel
rsh sagitta2 /tpc/tpcd/tpcd/custom/runstatsSet2.sh &
# do other tables
/tpc/tpcd/tpcd/custom/runstatsSet1.sh
```

B.26 runstatsSet1.sh

```
#!/bin/ksh
ofile=/tpc/tpcd/tpcd/tools/runstatsSet1.out
db2 connect to tpcd  > $ofile
db2 -tvf /tpc/tpcd/tpcd/custom/runstatsSet1.clp  >>  $ofile
db2 connect reset >> $ofile
db2 terminate     >> $ofile
```

B.27 runstatsSet2.sh

```
#!/bin/ksh
ofile=/tpc/tpcd/tpcd/tools/runstatsSet2.out
. /tpc/tpcd/.profile
db2 connect to tpcd  > $ofile
db2 -tvf /tpc/tpcd/tpcd/custom/runstatsSet2.clp  >>  $ofile
db2 connect reset >> $ofile
db2 terminate     >> $ofile
```

B.28 runstatsSet1.clp

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

```
    r_comment like statistics )
  AND  INDEXES ALL;
commit;
RUNSTATS ON TABLE TPCD.CUSTOMER  WITH DISTRIBUTION on
all columns
  and columns (
    c_name like statistics,
    c_address like statistics,
    c_phone like statistics,
    c_mktsegment like statistics,
    c_comment like statistics)
  AND  INDEXES ALL;
commit;
RUNSTATS ON TABLE TPCD.ORDERS    WITH DISTRIBUTION on
all columns
  and columns (
    o_orderstatus like statistics,
    o_orderpriority like statistics,
    o_clerk like statistics,
    o_comment like statistics)
  AND  INDEXES ALL;
commit;
RUNSTATS ON TABLE TPCD.LINEITEM  WITH DISTRIBUTION on
all columns
  and columns (
    l_returnflag like statistics,
    l_linestatus like statistics,
    l_shipinstruct like statistics,
    l_shipmode like statistics,
    l_comment like statistics)
  AND  INDEXES ALL;
COMMIT WORK;
```

B.29 runstatsSet2.clp

```
RUNSTATS ON TABLE TPCD.SUPPLIER  WITH DISTRIBUTION on
all columns
  and columns (
    s_name like statistics,
    s_address like statistics,
    s_phone like statistics,
    s_comment like statistics)
  AND  INDEXES ALL;
commit;
RUNSTATS ON TABLE TPCD.PART      WITH DISTRIBUTION on all
columns
  and columns (
    p_name like statistics,
    p_mfgr like statistics,
    p_brand like statistics,
    p_type like statistics,
    p_container like statistics,
    p_comment like statistics)
  AND  INDEXES ALL;
commit;
RUNSTATS ON TABLE TPCD.PARTSUPP  WITH DISTRIBUTION on
all columns
  and columns (
    ps_comment like statistics)
  AND  INDEXES ALL;
commit;
```

B.30 db2set_run.sh

```
# no change from load
```

B.31 dbmcfg_run.clp

```
-- no change from load
```

B.32 dbcfg_run

```
update database configuration for tpcd using
stat_heap_sz   4384
util_heap_sz   5000
locklist      32768;
```

B.33 tpcd.setup

```
TPCD_VERSION=2
TPCD_PLATFORM=aix
TPCD_DBNAME=TPCD
TPCD_WORKLOAD=H
TPCD_AUDIT_DIR=/tpc/tpcd/tpcd
TPCD_PRODUCT=v5
TPCD_MODE=mln
TPCD_PHYS_NODE=8
TPCD_LN_PER_PN=16
TPCD_SF=10000
TPCD_BUILD_STAGE=ALL
TPCD_DBPATH=/db2
TPCD_DDLPATH=/tpc/tpcd/tpcd/custom
TPCD_BUFFERPOOL_DEF=bp.ddl
TPCD_BUFFERPOOL_DEF_RUN=NULL
TPCD_NODEGROUP_DEF=ng.ddl
TPCD_EXPLAIN_DDL=explaintables.ddl
TPCD_TBSP_DDL=tbsp.ddl
TPCD_DDL=tbl.ddl
TPCD_QUAL_TBSP_DDL=NULL
TPCD_QUAL_DDL=NULL
TPCD_INDEXDDL=idx.ddl
TPCD_EXTRAINDEX=no
TPCD_ADD_RI=NULL
TPCD_AST=NULL
TPCD_RUNSTATS=runstats.sh
TPCD_RUNSTATSHORT=
TPCD_DBGEN=/tpc/tpcd/tpcd/appendix.v2/dbgen
TPCD_INPUT=/rawdata
TPCD_QUAL_INPUT=NULL
TPCD_TAILOR_DIR=/tmp/tpcd
TPCD_EARLYINDEX=no
TPCD_LOAD_DB2SET_SCRIPT=db2set_load.sh
TPCD_LOAD_CONFIGFILE=dbcfg_load.clp
TPCD_LOAD_DBM_CONFIGFILE=dbmcfg_load.clp
TPCD_LOAD_QUALCONFIGFILE=NULL
TPCD_LOAD_DBM_QUALCONFIGFILE=NULL
TPCD_LOADSTATS=no
TPCD_TEMP=
TPCD_SORTBUF=
TPCD_LOAD_PARALLELISM=0
TPCD_COPY_DIR=NULL
TPCD_FASTPARSE=yes
TPCD_BACKUP_DIR=
TPCD_LOGPRIMARY=NULL
TPCD_LOGFILSIZ=NULL
TPCD_LOGSECOND=NULL
TPCD_LOG_DIR=NULL
TPCD_LOG_DIR_SETUP_SCRIPT=NULL
TPCD_LOG_QUAL_DIR=NULL
TPCD_LOG=no
TPCD_DB2SET_SCRIPT=db2set_run.sh
TPCD_CONFIGFILE=dbcfg_run.clp
TPCD_DBM_CONFIG=dbmcfg_run.clp
TPCD_QUALCONFIGFILE=NULL
TPCD_DBM_QUALCONFIG=NULL
```

```
TPCD_MACHINE=big
TPCD_SMPDEGREE=1
TPCD_AGENTPRI=NULL
TPCD_ACTIVATE=no
TPCD_AUDIT=yes
TPCD_TMP_DIR=/tmp/tpcd/TEMP
TPCD_SHARED_TEMP_FULL_PATHNAME=/tpc/tpcd/tpcd/tmp
TPCD_QUERY_TEMPLATE_DIR=standard.V2
TPCD_QUAL_DBNAME=NULL
TPCD_NUMSTREAM=9
TPCD_FLATFILES=/db2dump/LINKS/uflinks
TPCD_STAGING_TABLE_DDL=UFtbl.ddl
TPCD_PRELOAD_STAGING_TABLE_SCRIPT=UFtbl_load.sh
TPCD_DELETE_STAGING_TABLE_SQL=UFtbl_delete.ddl
TPCD_UPDATE_IMPORT=false
TPCD_SPLIT_UPDATES=60
TPCD_CONCURRENT_INSERTS=30
TPCD_CONCURRENT_INSERTS_LOAD=1
TPCD_SPLIT_DELETES=60
TPCD_CONCURRENT_DELETES=30
TPCD_GEN_UPDATEPAIRS=32
TPCD_GENERATE_SEED_FILE=yes
TPCD_RUN_ON_MULTIPLE_NODES=NULL
TPCD_STATS_INTERVAL=5
TPCD_STATS_THRU_INT=60
TPCD_PS_INTERVAL=900
TPCD_PS_THRU_INT=1800
TPCD_GATHER_STATS=off
TPCD_UFTEMP=tabdata
TPCD_HAVECOMPILER=yes
TPCD_SLEEP=5
TPCD_INLISTMAX=default
TPCD_LOAD_SCRIPT=load.sh
TPCD_LOAD_SCRIPT_QUAL=NULL
TPCD_ROOTPRIV=yes
TPCD_DB2LOG=/db2dump/db2dump
TPCD_APPEND_ON=no
```

B.34 buildtpcd

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

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

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

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

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

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

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

```
if (length($ENV{"TPCD_STAGING_TABLE_DDL"}) <= 0)
{
   $ENV{"TPCD_STAGING_TABLE_DDL"} = "NULL";
}
if (length($ENV{"TPCD_PRELOAD_STAGING_TABLE_SCRIPT"}) <=
0)
{
   $ENV{"TPCD_PRELOAD_STAGING_TABLE_DDL"} = "NULL";
}
if (length($ENV{"TPCD_DELETE_STAGING_TABLE_SQL"}) <= 0)
{
   $ENV{"TPCD_DELETE_STAGING_TABLE_DDL"} = "NULL";
}
if (length($ENV{"TPCD_TBSP_DDL"}) <= 0)
{
   $ENV{"TPCD_TBSP_DDL"} = "dss.tbsp.ddl";
}
if (length($ENV{"TPCD_INDEXDDL"}) <= 0)
{
   $ENV{"TPCD_INDEXDDL"} = "dss.index";
}
if (length($ENV{"TPCD_RUNSTATS"}) <= 0)
{
   $ENV{"TPCD_RUNSTATS"} = "dss.runstats";
}
if (length($ENV{"TPCD_RUNSTATSHORT"}) <= 0)
{
   $ENV{"TPCD_RUNSTATSHORT"} = "NULL";
}
if (length($ENV{"TPCD_ADD_RI"}) <= 0)
{
   $ENV{"TPCD_ADD_RI"} = "NULL";
}
if (length($ENV{"TPCD_AST"}) <= 0)
{
   $ENV{"TPCD_AST"} = "NULL";
}
if ( ($ENV{"TPCD_INPUT"}) eq "NULL" )
{
   if (length($ENV{"TPCD_DBGEN"}) <= 0)
   {
      die "Must set TPCD_DBGEN if pregenerated flatfiles are not pro-
vided (TPCD_INPUT=NULL)\n";
   }
}
if ( $qual eq "QUAL" )
{
   if ( ($ENV{"TPCD_QUAL_INPUT"}) eq "NULL" )
   {
      if (length($ENV{"TPCD_DBGEN"}) <= 0)
      {
         die "Must set TPCD_DBGEN if pregenerated flatfiles are not
provided (TPCD_QUAL_INPUT=NULL)\n";
      }
}
}
if (length($ENV{"TPCD_TEMP"}) <= 1)
{
   $ENV{"TPCD_TEMP"} = "/u/$instance/sqllib/tmp";

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

```
    die "TPCD_AUDIT_DIR environment variable not set\n";
}
if (length($ENV{"TPCD_LOAD_DB2SET_SCRIPT"}) <= 0)
{
    $ENV{"TPCD_LOAD_DB2SET_SCRIPT"}="NULL"
}
if (length($ENV{"TPCD_DB2SET_SCRIPT"}) <= 0)
{
    $ENV{"TPCD_DB2SET_SCRIPT"}="NULL"
}
if (length($ENV{"TPCD_BUFFERPOOL_DEF"}) <= 0)
{
    $ENV{"TPCD_BUFFERPOOL_DEF"}="NULL"
}
if (length($ENV{"TPCD_EXPLAIN_DDL"}) <= 0)
{
    $ENV{"TPCD_EXPLAIN_DDL"}="NULL"
}
if (length($ENV{"TPCD_NODEGROUP_DEF"}) <= 0)
{
    $ENV{"TPCD_NODEGROUP_DEF"}="NULL"
}
if (length($ENV{"TPCD_PHYS_NODE"}) <= 0)
{
    $ENV{"TPCD_NODEGROUP_DEF"}="NULL"
}
if (length($ENV{"TPCD_APPEND_ON"}) <= 0)
{
    $ENV{"TPCD_APPEND_ON"}="yes"
}

#set up local variables
$tpcdVersion=$ENV{"TPCD_VERSION"};
$buildStage=$ENV{"TPCD_BUILD_STAGE"};
$product=$ENV{"TPCD_PRODUCT"};
$dbname=$ENV{"TPCD_DBNAME"};
$mode=$ENV{"TPCD_MODE"};
$sf=$ENV{"TPCD_SF"};
$sfReal=$sf;            # need a "saved" one for qualification stuff
$dbpath=$ENV{"TPCD_DBPATH"};
$ddlpath=$ENV{"TPCD_DDLPATH"};
$ddl=$ENV{"TPCD_DDL"};
$stagingTbl=$ENV{"TPCD_STAGING_TABLE_DDL"};
$preloadSam-
pleUF=$ENV{"TPCD_PRELOAD_STAGING_TABLE_SCRIPT"};
$deleteSampleUF=$ENV{"TPCD_DELETE_STAGING_TABLE_SQL"};
$buffpooldef=$ENV{"TPCD_BUFFERPOOL_DEF"};
$buffpooldef_run=$ENV{"TPCD_BUFFERPOOL_DEF_RUN"};
$nodegroupdef=$ENV{"TPCD_NODEGROUP_DEF"};
$tbspddl=$ENV{"TPCD_TBSP_DDL"};
$indexddl=$ENV{"TPCD_INDEXDDL"};
$extraindex=$ENV{"TPCD_EXTRAINDEX"};
$runstats=$ENV{"TPCD_RUNSTATS"};
$runstatShort=$ENV{"TPCD_RUNSTATSHORT"};
$dbgen=$ENV{"TPCD_DBGEN"};
$input=$ENV{"TPCD_INPUT"};
$earlyindex=$ENV{"TPCD_EARLYINDEX"};
$ldtemp=$ENV{"TPCD_TEMP"};
$sortbuf=$ENV{"TPCD_SORTBUF"};
$load_parallelism=$ENV{"TPCD_LOAD_PARALLELISM"};
$loadstats=$ENV{"TPCD_LOADSTATS"};
$copydir=$ENV{"TPCD_COPY_DIR"};
$fparse=$ENV{"TPCD_FASTPARSE"};
$addRI=$ENV{"TPCD_ADD_RI"};
$astFile=$ENV{"TPCD_AST"};
$genSeed=$ENV{"TPCD_GENERATE_SEED_FILE"};
$appendOn=$ENV{"TPCD_APPEND_ON"};

if ( $fparse eq "yes" )
{
    $fastparse="FASTPARSE";
}
else
{
    $fastparse=" ";
}
$backupdir=$ENV{"TPCD_BACKUP_DIR"};
$logprimary=$ENV{"TPCD_LOGPRIMARY"};
$logsecond=$ENV{"TPCD_LOGSECOND"};
$logfilsiz=$ENV{"TPCD_LOGFILSIZ"};
$log=$ENV{"TPCD_LOG"};
$logDir=$ENV{"TPCD_LOG_DIR"};
$logDirScript=$ENV{"TPCD_LOG_DIR_SETUP_SCRIPT"};
$machine=$ENV{"TPCD_MACHINE"};
$configfile=$ENV{"TPCD_CONFIGFILE"};
$dbmconfig=$ENV{"TPCD_DBM_CONFIG"};
$loadconfigfile=$ENV{"TPCD_LOAD_CONFIGFILE"};
$loadDBMconfig=$ENV{"TPCD_LOAD_DBM_CONFIGFILE"};
$smpdegree=$ENV{"TPCD_SMPDEGREE"};
$agentpri=$ENV{"TPCD_AGENTPRI"};
$activate=$ENV{"TPCD_ACTIVATE"};
$RealAudit=$ENV{"TPCD_AUDIT"};
$loadscript=$ENV{"TPCD_LOAD_SCRIPT"};
$auditDir=$ENV{"TPCD_AUDIT_DIR"};
$loadsetScript=$ENV{"TPCD_LOAD_DB2SET_SCRIPT"};
$setScript=$ENV{"TPCD_DB2SET_SCRIPT"};
$explainDDL=$ENV{"TPCD_EXPLAIN_DDL"};
$user=$ENV{"USER"};

# set up override of some parameters to override for qualification data-
base
if ( $qual eq "QUAL" )
{
    $loadscript=$ENV{"TPCD_LOAD_SCRIPT_QUAL"};
    if ( length($ENV{"TPCD_QUAL_DBNAME"}) <= 0 )
    {
        die "TPCD_QUAL_DBNAME environment variable not set\n";
    }
    $dbname=$ENV{"TPCD_QUAL_DBNAME"};

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

    $sf=$ENV{"TPCD_QUAL_SF"};
    if ( length($ENV{"TPCD_QUAL_DDL"}) <= 0 )
    {
        die "TPCD_QUAL_DDL environment variable not set\n";
    }
    $ddl=$ENV{"TPCD_QUAL_DDL"};
    if ( length($ENV{"TPCD_QUAL_TBSP_DDL"}) <= 0 )
    {
        die "TPCD_QUAL_TBSP_DDL environment variable not set\n";
    }
    $tbspddl=$ENV{"TPCD_QUAL_TBSP_DDL"};
    $input=$ENV{"TPCD_QUAL_INPUT"};
    if ( length($ENV{"TPCD_QUALCONFIGFILE"}) <= 0 )
    {
        die "TPCD_QUALCONFIGFILE environment variable not set\n";
    }
    $configfile=$ENV{"TPCD_QUALCONFIGFILE"};
    if ( length($ENV{"TPCD_DBM_QUALCONFIG"}) <= 0 )
    {
        die "TPCD_DBM_QUALCONFIG environment variable not set\n";
    }
    $dbmconfig=$ENV{"TPCD_DBM_QUALCONFIG"};
    if ( length($ENV{"TPCD_LOAD_QUALCONFIGFILE"}) <= 0 )
    {
        die "TPCD_LOAD_QUALCONFIGFILE environment variable not
set\n";
    }
    $loadconfigfile=$ENV{"TPCD_LOAD_QUALCONFIGFILE"};
```

```
    if ( length($ENV{"TPCD_LOAD_DBM_QUALCONFIGFILE"}) <= 0 )
    {
       die "TPCD_LOAD_DBM_QUALCONFIGFILE environment vari-
able not set\n";
    }
    $loadDBMconfig=$ENV{"TPCD_LOAD_DBM_QUALCONFIGFILE"};
    if (length($ENV{"TPCD_LOG_DIR"}) <= 0)
    {
       $ENV{"TPCD_LOG_DIR"} = "NULL";
    }
    $logDir=$ENV{"TPCD_LOG_QUAL_DIR"};
}

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

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

  if ( $machine eq "small" )
  {
     $buffpage = 5000;
     $sortheap = 3000;
     $sheapthres = 8000;
     $util_heap_sz = 5000;
     $ioservers = 6;
  }
  elsif ( $machine eq "medium" )
  {
     $buffpage = 100000;
     $sortheap =  8000;
     $sheapthres = 20000;
     $util_heap_sz = 100000;
     $ioservers = 10;
  }
  elsif ( $machine eq "big" )
  {
     $buffpage = 30000;
     $sortheap = 20000;
     $sheapthres = 50000;
     $util_heap_sz = 30000;
     $ioservers = 20;
  }
  elsif ( $machine eq "sunsmp" )
  {
     $buffpage = 60000;
     $sortheap = 20000;
     $sheapthres = 80000;
     $util_heap_sz = 30000;
     $ioservers = 80;
  }
  elsif ( $machine eq "eastwood" )
  {
     $buffpage = 80000;
     $sortheap = 50000;
     $sheapthres = 810000;
     $ioclnrs = 4;
     $chngpgs = 30;
     $ioservers = 21;
     $util_heap_sz = 50000;
  }
}
# echo parameter settings to acknowledge what is being built
if ( $buildStage ne "ALL" )
{
    print " ***** STARTING the build process at the $buildStage Stage
*****\n";
}
print "Building a TPC-D Version $tpcdVersion $sf GB database on
$dbpath with: \n";
print "   Mode = $mode \n";
print "   Tablespace ddl in $ddlpath${delim}$tbspddl \n";
if ( $nodegroupdef ne "NULL" )
{
   print "   Nodegroup ddl in $ddlpath${delim}$nodegroupdef \n";
}
if ( $buffpooldef ne "NULL" )
{
   print "   Bufferpool ddl in $ddlpath${delim}$buffpooldef \n";
}
if ( $buffpooldef_run ne "NULL" )
{
   print "   Run Bufferpool ddl in $ddlpath${delim}$buffpooldef_run \n";
}
print "   Table ddl in $ddlpath${delim}$ddl \n";
print "   Index ddl in              $ddlpath${delim}$indexddl\n";
if ( $extraindex ne "no" )
{
  print " Indices to create after the load $ddlpath${delim}$extraindex\n";
}
if ( $loadscript eq "NULL")
{
    if ( $input eq "NULL" )
    {
       print "   Data generated by DBGEN in $dbgen\n";
    }
    else
    {
       print "   Data loaded from flat files in $input\n";
    }
}
if ( $earlyindex eq "yes" )
{
  print "   Indexes created before loading\n";
}
else
{
  print "   Indexes created after loading\n";
}
if ( $addRI ne "NULL" )
{
    print "   RI being used from $ddlpath${delim}$addRI\n";
}
if ( $astFile ne "NULL" )
{
    print "   AST being used from $ddlpath${delim}$astFile\n";
}
if ( $loadstats eq "yes" )
{
  if ( $earlyindex eq "yes" )
  {
    print "   Statistics for tables and indexes gathered during load\n";
```

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

if (($loadconfigfile eq "") || ($loadconfigfile eq "NULL"))
{
 print "   Machine size set to $machine so the following configura-
tion\n";
 print "   parameters are used for load, create index and runstats: \n";
 print "      BUFFPAGE = $buffpage \n";
 print "      SORTHEAP = $sortheap \n";
 print "      SHEAPTHRES = $sheapthres\n";
 print "      NUM_IOSERVERS = $ioservers\n";
 print "      NUM_IOCLEANERS = $ioclnrs\n";
 print "      CHNGPGS_THRESH = $chngpgs\n";
 print "      UTIL_HEAP_SZ = $util_heap_sz\n";
 print "      Degree of parallelism (dft_degree and max_querydegree)
set to $smpdegree\n";
 print "      Parameters for load are: temp file      = $ldtemp\n";
 print "                               sort buf       = $sortbuf\n";
 print "                               ld parallelism = $load_parallelism\n";
 if ( $fparse eq "yes" )
 {
  print "                                FASTPARSE used on load\n";
 }
}
if ( $loadscript ne "NULL")
{
 print "   Load commands  in $ddlpath${delim}$loadscript\n";
}



print "   Degree of parallelism (dft_degree and max_querydegree) set
to $smpdegree\n";
if ( $agentpri ne "NULL" )

{
   print "   AGENTPRI set to $agentpri\n";
}
if ( $activate eq "yes" )
```

```
{
   print "   Database will be activated when build is complete\n";
}
if ( $explainDDL ne "NULL" )
{
   print "   EXPLAIN DDL being used from
$ddlpath${delim}$explainDDL\n";
}
else
{
   print "   EXPLAIN DDL being used from default sqllib directory\n";
}

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

# don't need separate calls for mln/mpp vs uni since the $all_ln will be
# defined appropriately
if ( $platform eq "nt" )
{
   if (($mode eq "uni") || ($mode eq "smp"))
   {
     #spaces required for NT
     $rc=&dodb_noconn("db2set DB2OPTIONS=\"   -t -v +c\"",$all_ln);
   }
   else
   {
     $rc=&dodb_noconn("db2set DB2OPTIONS=\\\"   -t -v
+c\\\"",$all_ln);
   }
}
else
{
   if (($mode eq "uni") || ($mode eq "smp"))
   {
     $rc=&dodb_noconn("db2set DB2OPTIONS=\"-t -v +c\"",$once);
   }
   else
   {
     $rc=&dodb_noconn("db2set DB2OPTIONS=\\\"-t -v +c\\\"",$once);
   }
}

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

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

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

# set the db2 env vars for loading, from the
TPCD_LOAD_DB2SET_SCRIPT script
if ( $loadsetScript ne "NULL" )
{
   if ( $platform eq "nt" )
   {
      #### Mike , th, the- for some reason rah on NT doesn't like run-
ning
      #### a fully qualified file name.  Switch to dir, then run
      ## check CHANGE THIS to have single node and multinode exe-
cution
      if (( $mode eq "uni" ) || ( $mode eq "smp" ))

      {
              system("db2set -r $instance");
         $ret=system("${ddlpath}${delim}$loadsetScript");
         #$ret=system("cd ${ddlpath} $sep $loadsetScript");
      }
      else
      {
              system(" rah \" db2set -r $instance\" ");
         $ret=system(" rah \" cd ${ddlpath} & $loadsetScript\" ");
      }
   }
   else
   {
           system("db2set -r $instance");
      $ret=system("${ddlpath}${delim}$loadsetScript");
   }
   if ( $ret != 0 )
   {
      die "failure setting load time db2set parms from $loadsetScript \n";
   }
}
# stopping and starting db2 before we continue
print "Stopping DB2 ...\n";
$rc=system("db2stop");
if ( $rc < 0 )
{
   die "failure during db2stop rc = $rc \n";
}
print "Starting DB2 ...\n";
$rc=system("db2start");
#if ( $rc != 0 )
#{
#    die "failure during db2start rc = $rc \n";
#}

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

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

   # reset the dbm configuration before we start
   # and drop deadlock event monitor (kwai 20030721)

   &dodb_conn($dbname,"db2 grant connect on database to public
$sep      \
     db2 commit $sep  \
     db2 set event monitor db2detaildeadlock state 0 $sep \
     db2 commit $sep  \
     db2 drop event monitor db2detaildeadlock $sep \
     db2 commit",$once);

   &dodb_noconn("db2 reset database manager configuration",$once);

   # update the log information first
   # set up the log directory before we do any index creation
   if ($logDirScript ne "NULL")
   {
     system ("$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 ( $setLogs != 0 )
   {
     &dodb_noconn("$setLogString",$all_ln);
   }

   # setup the load configuration
   &outtime("*** Setting LOAD configuration.");
   if (($loadconfigfile eq "") || ($loadconfigfile eq "NULL"))
   {
     &dodb_noconn("db2 update db cfg for $dbname using buffpage
$buffpage $sep \
          db2 update db cfg for $dbname using sortheap $sortheap $sep
\
          db2 update db cfg for $dbname using num_iocleaners $ioclnrs
$sep \
          db2 update db cfg for $dbname using num_ioservers
$ioservers $sep \
          db2 update db cfg for $dbname using util_heap_sz
$util_heap_sz $sep \
          db2 update db cfg for $dbname using chngpgs_thresh
$chngpgs",
          $all_ln);
     &dodb_noconn("db2 update dbm cfg using sheapthres $sheap-
thres",$once);
   }
   else
   {
      &dodb2file_noconn("$ddlpath${delim}$loadconfigfile",$all_ln);
      &dodb2file_noconn("$ddlpath${delim}$loadDBMconfig",$once);
   }
   system(" db2_all \"||\\\" db2 get db cfg for tpcd >
/db2dump/dbcfg_load.## \" ");
   &dodb_noconn("db2 terminate",$once);
   $rc=system("db2stop");
   if ( $rc != 0 )
   {
      die "failure during db2stop after resetting for load config rc = $rc
\n";
   }
   $rc=system("db2start");
   if ( $rc != 0 )
   {
      die "failure during db2start rc = $rc \n";
   }
} # end of "CREATE DATABASE" phase
if (( $buildStage eq "ALL" ) || ( $buildStage eq "CRTTBSP" ) ||
   ( $buildStage eq "LOAD" ) ||
   (( $buildStage eq "INDEX" ) && ( $earlyindex eq "yes" )) )
{
   # create the nodegroups is there is a file specified
   if ( $nodegroupdef ne "NULL" )
   {
      #run the create bufferpool ddl
      &outtime("*** Creating the nodegroups");
      &dodb2file($dbname,"$ddlpath${delim}$nodegroupdef",$once);

   }

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

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

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

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

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

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

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

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

    if ( $mode eq "mpp" )
    {
       #need parallel specific
       print "need to figure parallel specific creation of tmp\n";
    }
    mkdir("${delim}tmp${delim}$instance",0777);
}  # end of "CREATE TABLESPACE and TABLES" phase
if (( $buildStage eq "ALL" ) || ( $buildStage eq "CRTTBSP" ) ||
    ( $buildStage eq "LOAD" ) ||
    (( $buildStage eq "INDEX" ) && ( $earlyindex eq "yes" )) )
{
    # do the load stage of the build, or if early index is on do
    # the index creation also
    # setup the load configuration
    if ( $buildStage ne "ALL")
    { # we're restarting a build so reapply the load config
      &outtime("*** Setting LOAD configuration.");
      if (($loadconfigfile eq "") || ($loadconfigfile eq "NULL"))
      {
       &dodb_noconn("db2 update db cfg for $dbname using buffpage
$buffpage $sep \
            db2 update db cfg for $dbname using sortheap $sortheap $sep
\
            db2 update db cfg for $dbname using num_iocleaners $ioclnrs
$sep \
            db2 update db cfg for $dbname using num_ioservers
$ioservers $sep \
            db2 update db cfg for $dbname using util_heap_sz
$util_heap_sz $sep \
            db2 update db cfg for $dbname using chngpgs_thresh
$chngpgs",
            $all_ln);
       &dodb_noconn("db2 update dbm cfg using sheapthres $sheap-
thres",$once);
      }
      else
      {
       &dodb2file_noconn("$ddlpath${delim}$loadconfigfile",$all_ln);
       &dodb2file_noconn("$ddlpath${delim}$loadDBMconfig",$once);
      }
      &dodb_noconn("db2 terminate",$once);
      $rc=system("db2stop");
      if ( $rc != 0 )
      {
        die "failure during db2stop after resetting for load config rc = $rc
\n";
      }
      $rc=system("db2start");
      if ( $rc != 0 )
      {
        die "failure during db2start rc = $rc \n";
      }
    }

    # if earlyindex requested, create indexes
    if ( $earlyindex eq "yes" )
    {
       &outtime("*** Starting to create indexes");
       &dodb2file($dbname,"$ddlpath${delim}$indexddl",$once);

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


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

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

       if ( $rc != 0 )
       {
          die "doload for $dbname failed rc = $rc. Script is
$ddlpath${delim}$loadscript\n";
       }
    }

    else
    {
```

```
      die "TPCD_MODE not set to one of uni, smp, mln or mpp\n";
   }

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


   # verify that the audit directory exists
   $filename="$auditDir";
   if (-e $filename)
   {
    # set up the $auditDir/$dbname.$user.update.pair.num file
    # to start at update pair 1
    $filename="$auditDir${delim}$dbname.$user.update.pair.num";
   }
   else
   {
    mkdir ("$auditDir", 0775) || die "cannot mkdir $auditDir";
   }
   print "setting update pair num to 1\n";
   system("echo 1 > $filename");

} # end all and load stage (and create index if early index was specified

if (( $buildStage eq "ALL" ) || ( $buildStage eq "CRTTBSP" ) ||
   ( $buildStage eq "LOAD" ) ||
   ( $buildStage eq "INDEX" ))
{
   if ( $buildStage eq "INDEX")
   { # we're restarting a build so reapply the load config
     &outtime("*** Setting LOAD configuration.");
     if (($loadconfigfile eq "") || ($loadconfigfile eq "NULL"))
     {
      &dodb_noconn("db2 update db cfg for $dbname using buffpage
$buffpage $sep \
          db2 update db cfg for $dbname using sortheap $sortheap $sep
\
          db2 update db cfg for $dbname using num_iocleaners $ioclnrs
$sep \
          db2 update db cfg for $dbname using num_ioservers
$ioservers $sep \
          db2 update db cfg for $dbname using util_heap_sz
$util_heap_sz $sep \
          db2 update db cfg for $dbname using chngpgs_thresh
$chngpgs",
          $all_ln);
      &dodb_noconn("db2 update dbm cfg using sheapthres
$sheapthres",$once);
     }
     else
     {
       &dodb2file_noconn("$ddlpath${delim}$loadconfigfile",$all_ln);
       &dodb2file_noconn("$ddlpath${delim}$loadDBMconfig",$once);
     }
     &dodb_noconn("db2 terminate",$once);
     $rc=system("db2stop");
     if ( $rc != 0 )
     {
       die "failure during db2stop after resetting for load config rc = $rc
\n";
     }
     $rc=system("db2start");
     if ( $rc != 0 )
     {
       die "failure during db2start rc = $rc \n";
     }
   }
   # if indexes haven't been created, do so now
   if ( $earlyindex ne "yes" )
   {
       &outtime("*** Create index started");
       &dodb2file($dbname,"$ddlpath${delim}$indexddl",$once);

       &outtime("*** Create index completed");
   }
   if ( $extraindex ne "no" )
   {
       # use this additional file for indexes
       &outtime("*** Create index (part 2) started");
       &dodb2file($dbname,"$ddlpath${delim}$extraindex",$once);
       &outtime("*** Create index (part 2) completed");
   }

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

if (( $buildStage eq "ALL" ) ||  ( $buildStage eq "CRTTBSP" ) ||
   ( $buildStage eq "LOAD" ) ||
   ( $buildStage eq "INDEX" ) || ( $buildStage eq "RUNSTATS" ))
{
   if ( $buildStage eq "RUNSTATS")
   { # we're restarting a build so reapply the load config
     &outtime("*** Setting LOAD configuration.");
     if (($loadconfigfile eq "") || ($loadconfigfile eq "NULL"))
     {
      &dodb_noconn("db2 update db cfg for $dbname using buffpage
$buffpage $sep \
          db2 update db cfg for $dbname using sortheap $sortheap $sep
\
          db2 update db cfg for $dbname using num_iocleaners $ioclnrs
$sep \
          db2 update db cfg for $dbname using num_ioservers
$ioservers $sep \
          db2 update db cfg for $dbname using util_heap_sz
$util_heap_sz $sep \
          db2 update db cfg for $dbname using chngpgs_thresh
$chngpgs",
          $all_ln);
      &dodb_noconn("db2 update dbm cfg using sheapthres $sheap-
thres",$once);
     }
     else
     {
       &dodb2file_noconn("$ddlpath${delim}$loadconfigfile",$all_ln);
       &dodb2file_noconn("$ddlpath${delim}$loadDBMconfig",$once);
     }
     &dodb_noconn("db2 terminate",$once);
     $rc=system("db2stop");
     if ( $rc != 0 )
     {
       die "failure during db2stop after resetting for load config rc = $rc
\n";
     }
     $rc=system("db2start");
     if ( $rc != 0 )
     {
       die "failure during db2start rc = $rc \n";
     }
   }
   # 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 dur-
ing load,
   # or after load)
   # We need to run the runstats as well if we have specifed an extra
index file
   # for "after load" indexes
   if (( $loadstats eq "no" ) || ( $earlyindex eq "no" ) || ( $extraindex ne
"no" ))
   {
       # if loadstats not gathered, then index stats not gathered either.
```

```
      &outtime("*** Runstats started");

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

&dodb2file($dbname,"$ddlpath${delim}$runstatShort",$once);
         }
      }
      # run the full runstats, or the remainder of what wasn't put into the
short
      # runstats file. You should be sure that this runstats will take
longer
      # than the short runstats that is running in the background, other-
wise
      # setting the config will happen before this is done.
      system("$ddlpath${delim}$runstats");
      &outtime("*** Runstats completed");
   }
} # end build phaose all/load/index/runstats

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

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

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

# create the explain tables (reminder explain tables should be on cata-
log partition for speed)
if ( $explainDDL ne "NULL" )
{
   $explnPathFile="$ddlpath${delim}$explainDDL";
}
else
{
   if ( $platform eq "ptx" )
   {
      $home=$ENV{"HOME"};
      $sqlpath="$home${delim}sqllib";
   }
   if ( $platform ne "nt" )
   {
      $sqlpath="~${delim}sqllib";
   }
   else
   {
      $sqlpath=$ENV{"DB2PATH"};
   }
   $explnPathFile="$sqlpath${delim}misc${delim}EXPLAIN.DDL";
}
&outtime("*** Adding explaintables started");
&dodb2file($dbname,"$explnPathFile",$once);

# set the configuration
&outtime("*** Set Configuration started");
&dodb2file_noconn("${ddlpath}${delim}$configfile",$all_ln);
&dodb2file_noconn("${ddlpath}${delim}$dbmconfig",$once);
system(" db2_all \"||\\\" db2 get db cfg for tpcd >
/db2dump/dbcfg_run.## \" ");

if ( $buffpooldef_run ne "NULL" )
{
   # set run bufferpools   kmw-ironwood
   &outtime("*** Altering the bufferpools");
   &dodb2file($dbname,"$ddlpath${delim}$buffpooldef_run",$once);
}

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

# set the db2 environment variables for running the benchmark
if ( $setScript ne "NULL" )
{
   if ( $platform eq "aix" || $platform eq "sun" || $platform eq "ptx")
   {
            system("db2set -r $instance");
      $ret=system("${ddlpath}${delim}$setScript");
   }
   elsif ( $platform eq "nt" )
   {
      if (($mode eq "uni" ) || ($mode eq "smp" ))
      {
         system("db2set -r $instance");
         $ret = system("perl ${ddlpath}${delim}$setScript");
      }
      else
      {
         system(" rah \" db2set -r $instance\" ");
         $ret = system(" rah \" cd ${ddlpath} & $setScript\" ");
      }
   }
   if ( $ret != 0 )
   {
      die "failure setting runtime db2set parms from $setScript \n";
   }
}
# if logging is enabled, we must take a backup of the database
if ( $log eq "yes" )
{
   &dodb_noconn("db2 update database configuration for $dbname us-
ing LOGRETAIN yes",$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\" \' ");
         if ( $rc != 0 )
         {
            die "backup of catalog node failed rc = $rc\n";
         }
         # back up remaining nodes
         $rc=system("db2_all \'||}]<<-000< db2 backup database
$dbname to $backupdir without prompting\' ");
         if ( $rc != 0 )
         {
            die "backup of remaining nodes failed rc = $rc\n";
         }
      }
      else
      {
         &dodb_noconn("db2 backup database $dbname to $backup-
dir",$once);
      }
      &outtime("*** Finished the backup");

   }
   else
   {
      # This is the test database. Clause 3.1.4 states that "the test
sponsor is
      # not required to make or have backup copies of the test data-
base; however
      # all other mechanisms that guarantee durability of the qualifica-
tion
      # database must be enabled in the same way for the test data-
base".
      # According to this clause we do need to keep the backup of the
database.
      &dodb_noconn("db2dart $dbname /CHST /WHAT DBBP
OFF",$all_ln);
   }
}

# deactivate the database
&dodb_noconn("db2 deactivate database $dbname",$once);
# stop and restart the database to get config parms recognized
$rc=system("db2stop");
if ( $rc != 0 )
{
   die "failure during db2stop rc = $rc \n";
}
$rc=system("db2start");
if ( $rc != 0 )
{
   die "failure during db2start rc = $rc \n";
}

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

#create generated seeds
if ( $genSeed ne "no" )
{
  $rc = system("perl createmseedme.pl 10000");
  if ( $rc != 0 )
  {
    warn "createmseedme failed\n";
  }
}

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

if ( $RealAudit eq "yes" )
{
   # if we are in real audit mode then we have to do a number of things
   # set up the audit directory structure and the run directory structure
   # so that once we have completed the buildtpcd, we are ready to
run.
   # first remove any old "update pair number" file so we won't get
prompted
   # doing setupDir
   &rm("$auditDir${delim}tools${delim}tpcd.runsetup");
   system("perl setupRun");
   # before we stop the database for the final time
   # if we are in the real audit mode then run dbtables and dbcheck be-
fore
   # we print out the final notice that we are ready to run the perform-
ance tests
   # if we are building the qualification database then we will bind to
both
   # the dbname database and the qualification database
   if ( $qual eq "QUAL" )
   {
      $verifyType="q";
   }
   else
   {
      $verifyType="t";
   }
   system("perl tablesdb $verifyType");

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

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

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

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

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

if ( $RealAudit ne "yes" )
{
   # if we are not in a real audit, then we can restart the database
manager
   # if we are in a real audit, then we don't want to do this until the
   # power test starts
   $rc=system("db2start");
   if ( $rc != 0 )
   {
      die "failure during db2start rc = $rc \n";
   }
   if ( $activate eq "yes" )
   {
      &dodb_noconn("db2 activate database $dbname",$once);
   }
}

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

1;
```

Appendix - C Query Text and Output

C.1 Qualification Query Output

Query 1

```
-- 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_big(*) as count_order
from
tpcd.lineitem
where
l_shipdate <= date ('1998-12-01') - 90 day
group by
l_returnflag,
l_linestatus
order by
l_returnflag,
l_linestatus

L_RETURNFLAG  L_LINESTATUS  SUM_QTY
SUM_BASE_PRICE       SUM_DISC_PRICE       SUM_CHARGE
AVG_QTY              AVG_PRICE            AVG_DISC
COUNT_ORDER
---------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------
------------------------
A            F                37734107.000      56586554400.730
53758257134.870      55909065222.828              25.522
38273.130            0.050                    1478493.
N            F                  991417.000      1487504710.380
1413082168.054       1469649223.194               25.516
38284.468            0.050                     38854.
N            O                74476040.000      111701729697.740
106118230307.605     110367043872.497             25.502
38249.118            0.050                    2920374.
R            F                37719753.000      56568041380.900
53741292684.604      55889619119.832              25.506
38250.855            0.050                    1478870.


Number of rows retrieved is:      4
```

Query 2

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

Tag: Q2     Stream: -1   Sequence number: 2

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

S_ACCTBAL            S_NAME                    N_NAME
P_PARTKEY    P_MFGR                    S_ADDRESS
S_PHONE          S_COMMENT
---------------------------------------------------------------------------------------------
---------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------
          9938.530  Supplier#000005359         UNITED KINGDOM
185358  Manufacturer#4             QKuHYh,vZGiwu2FWEJoLDx04
33-429-790-6131  blithely silent pinto beans are furiously. slyly final
deposits acros
          9937.840  Supplier#000005969         ROMANIA
108438  Manufacturer#1
ANDENSOSmk,miq23Xfb5RWt6dvUcvt6Qa          29-520-692-3537
carefully slow deposits use furiously. slyly ironic platelets above the
ironic
          9936.220  Supplier#000005250         UNITED KINGDOM
249  Manufacturer#4             B3rqp0xbSEim4Mpy2RH J
33-320-228-2957  blithely special packages are. stealthily express
deposits across the closely final instructi
          9923.770  Supplier#000002324         GERMANY
29821  Manufacturer#4             y3OD9UywSTOk                    17-
779-299-1839  quickly express packages breach quiet pinto beans.
requ
          9871.220  Supplier#000006373         GERMANY
43868  Manufacturer#5             J8fcXWsTqM                      17-
813-485-8637  never silent deposits integrate furiously blit
          9870.780  Supplier#000001286         GERMANY
81285  Manufacturer#2
YKA,E2fjiVd7eUrzp2Ef8j1QxGo2DFnosaTEH       17-516-924-4574
final theodolites cajole slyly special,
```

9870.780 Supplier#000001286 GERMANY 181285 Manufacturer#4 YKA,E2fjiVd7eUrzp2Ef8j1QxGo2DFnosaTEH 17-516-924-4574 final theodolites cajole slyly special,
9852.520 Supplier#000008973 RUSSIA 18972 Manufacturer#2 t5L67YdBYYH6o,Vz24jpDyQ9 32-188-594-7038 quickly regular instructions wake-- carefully unusual braids into the expres
9847.830 Supplier#000008097 RUSSIA 130557 Manufacturer#2 xMe97bpE69NzdwLoX 32-375-640-3593 slyly regular dependencies sleep slyly furiously express dep
9847.570 Supplier#000006345 FRANCE 86344 Manufacturer#1 VSt3rzk3qG698u6ld8HhOByvrTcSTSvQlDQDag 16-886-766-7945 silent pinto beans should have to snooze carefully along the final reques
9847.570 Supplier#000006345 FRANCE 173827 Manufacturer#2 VSt3rzk3qG698u6ld8HhOByvrTcSTSvQlDQDag 16-886-766-7945 silent pinto beans should have to snooze carefully along the final reques
9836.930 Supplier#000007342 RUSSIA 4841 Manufacturer#4 JOlK7C1,7xrEZSSOw 32-399-414-5385 final accounts haggle. bold accounts are furiously dugouts. furiously silent asymptotes are slyly
9817.100 Supplier#000002352 RUSSIA 124815 Manufacturer#2 4LfoHUZjgjEbAKw TgdKcgOc4D4uCYw 32-551-831-1437 blithely pending packages across the ironic accounts grow slyly after the furiously
9817.100 Supplier#000002352 RUSSIA 152351 Manufacturer#3 4LfoHUZjgjEbAKw TgdKcgOc4D4uCYw 32-551-831-1437 blithely pending packages across the ironic accounts grow slyly after the furiously
9739.860 Supplier#000003384 FRANCE 138357 Manufacturer#2 o,Z3v4POifevE k9U1b 6J1ucX,I 16-494-913-5925 slyly ironic theodolites hag
9721.950 Supplier#000008757 UNITED KINGDOM 156241 Manufacturer#3 Atg6GnM4dT2 33-821-407-2995 ironic, even dolphins above the furiously ironic foxes sleep slyly around the caref
9681.330 Supplier#000008406 RUSSIA 78405 Manufacturer#1 ,qUuXcftUl 32-139-873-8571 furiously even deposits affix thinly special theodolites. furiou
9643.550 Supplier#000005148 ROMANIA 107617 Manufacturer#1 kT4ciVFslx9z4s79p Js825 29-252-617-4850 doggedly even ideas boost furiously against the furiously express
9624.820 Supplier#000001816 FRANCE 34306 Manufacturer#3 e7vab91vLJPWxxZnewmnDBpDmxYHrb 16-392-237-6726 blithely regular accounts cajole furiously. regular
9624.780 Supplier#000009658 ROMANIA 189657 Manufacturer#1 oE9uBgEfSS4opIcepXyAYM,x 29-748-876-2014 regular deposits haggle. furiously express asympto
9612.940 Supplier#000003228 ROMANIA 120715 Manufacturer#2 KDdpNKN3cWu7ZSrbdqp7AfSLxx,qWB 29-325-784-8187 carefully pending accounts serve. furiously close deposits boost slyly. q
9612.940 Supplier#000003228 ROMANIA 198189 Manufacturer#4 KDdpNKN3cWu7ZSrbdqp7AfSLxx,qWB 29-325-784-8187 carefully pending accounts serve. furiously close deposits boost slyly. q
9571.830 Supplier#000004305 ROMANIA 179270 Manufacturer#2 qNHZ7WmCzygwMPRDO9Ps 29-973-481-1831 furiously final deposits
9558.100 Supplier#000003532 UNITED KINGDOM 88515 Manufacturer#4 EOeuiiOn21OVpTlGguufFDFsbN1p0lhpxHp 33-152-301-2164 daring, sly accounts breach about th
9492.790 Supplier#000005975 GERMANY 25974 Manufacturer#5 S6mIiCTx82z7lV 17-992-579-4839 always pending packages boost slyly.
9461.050 Supplier#000002536 UNITED KINGDOM 20033 Manufacturer#1 8mmGbyzaU 7ZS2wJumTibypncu9pNkDc4FYA 33-556-973-5522 even foxes are quickly furiously express requests. packages
9453.010 Supplier#000000802 ROMANIA 175767 Manufacturer#1 ,6HYXb4uaHITmtMBj4Ak57Pd 29-342-882-6463 final, regular packages across the slowly regular packag
9408.650 Supplier#000007772 UNITED KINGDOM 117771 Manufacturer#4 AiC5YAH,gdu0i7 33-152-491-1126 blithely final ideas sleep carefully. requests are
9359.610 Supplier#000004856 ROMANIA 62349 Manufacturer#5 HYogcF3Jb yh1 29-334-870-9731 carefully unusual packages sleep carefully even ideas. dogged accoun
9357.450 Supplier#000006188 UNITED KINGDOM 138648 Manufacturer#1 g801,ssP8wpTk4Hm 33-583-607-1633 carefully regular deposits wake carefully furiously even i
9352.040 Supplier#000003439 GERMANY 170921 Manufacturer#4 qYPDgoiBGhCYxjgC 17-128-996-4650 fluffily regular pinto beans wake. unusual, final ideas c
9312.970 Supplier#000007807 RUSSIA 90279 Manufacturer#5 oGYMPCk9XHGB2PBfKRnHA 32-673-872-5854 unusual asymptotes above the
9312.970 Supplier#000007807 RUSSIA 100276 Manufacturer#5 oGYMPCk9XHGB2PBfKRnHA 32-673-872-5854 unusual asymptotes above the
9280.270 Supplier#000007194 ROMANIA 47193 Manufacturer#3 zhRUQkBSrFYxIAXTfInj vyGRQjeK 29-318-454-2133 slyly ironic requests despite the unusual ins
9274.800 Supplier#000008854 RUSSIA 76346 Manufacturer#3 1xhLoOUM7I3mZ1mKnerw OSqdbb4QbGa 32-524-148-5221 ruthlessly ironic instructions along the regular, furious requests integrate car
9249.350 Supplier#000003973 FRANCE 26466 Manufacturer#1 d18GiDsL6Wm2IsGXM,RZf1jCsgZAOjNYVThTRP4 16-722-866-1658 quickly ironic sauternes use b
9249.350 Supplier#000003973 FRANCE 33972 Manufacturer#1 d18GiDsL6Wm2IsGXM,RZf1jCsgZAOjNYVThTRP4 16-722-866-1658 quickly ironic sauternes use b
9208.700 Supplier#000007769 ROMANIA 40256 Manufacturer#5 rsimdze 5o9P Ht7xS 29-964-424-9649 furiously ruthless epitaphs among the furiously regular accounts use slowly fluffily ev
9201.470 Supplier#000009690 UNITED KINGDOM 67183 Manufacturer#5 CB BnUTlmi5zdeEl7R7 33-121-267-9529 blithely unusual accounts integrate slyly. platelets
9192.100 Supplier#000000115 UNITED KINGDOM 85098 Manufacturer#3 nJ 2t0f7Ve,wL1,6WzGBJLNBUCKlsV 33-597-248-1220 slyly bold pinto beans boost across the furiously regular packages. carefully regu
9189.980 Supplier#000001226 GERMANY 21225 Manufacturer#4 qsLCqSvLyZfuXlpjz 17-725-903-1381 final, express instruction
9128.970 Supplier#000004311 RUSSIA 146768 Manufacturer#5 l8IjnXd7NSJRs594RxsRR0 32-155-440-7120 regular pinto beans sleep ca
9104.830 Supplier#000008520 GERMANY 150974 Manufacturer#4 RqRVDgD0ER J9 b41vR2,3 17-728-804-1793 deposits sleep carefully e
9101.000 Supplier#000005791 ROMANIA 128254 Manufacturer#5 zub2zCV,jhHPPQqi,P2INAjE1zI n66cOEoXFG 29-549-251-5384 carefully ironic packages after the

9094.570 Supplier#000004582 RUSSIA
39575 Manufacturer#1 WB0XkCSG3r,mnQ
n,h9VIxjjr9ARHFvKgMDf 32-587-577-1351 asymptotes above the
slyly even requests haggle furiously about the regular accounts
8996.870 Supplier#000004702 FRANCE
102191 Manufacturer#5 8XVcQK23akp 16-
811-269-8946 stealthy requests haggle c
8996.140 Supplier#000009814 ROMANIA
139813 Manufacturer#2
af0O5pg83lPU4IDVmEylXZVqYZQzSDlYLAmR 29-995-571-8781
ironic theodolites are evenly unusual requests-- pending pinto beans
across the in
8968.420 Supplier#0000100000 ROMANIA
119999 Manufacturer#5 aTGLEusCiL4F
PDBdv665XBJhPyCOB0i 29-578-432-2146 furiously final ideas
believe furiously. furiously final ideas
8936.820 Supplier#000007043 UNITED KINGDOM
109512 Manufacturer#1
FVajceZInZdbJE6Z9XsRUxrUEpiwHDrOXi,1Rz 33-784-177-8208
furiously regular excuses wake after the blithely special pinto beans?
even instructions sl
8929.420 Supplier#000008770 FRANCE
173735 Manufacturer#4 R7cG26TtXrHAP9 HckhfRi
16-242-746-9248 final accounts sleep furiously. blithely ironic foxes
wake boldly across the furiously s
8920.590 Supplier#000003967 ROMANIA
26460 Manufacturer#1 eHoAXe62SY9 29-
194-731-3944 quickly even requests should have to affix blithely-- fur
8920.590 Supplier#000003967 ROMANIA
173966 Manufacturer#2 eHoAXe62SY9 29-
194-731-3944 quickly even requests should have to affix blithely-- fur
8913.960 Supplier#000004603 UNITED KINGDOM
137063 Manufacturer#2
OUzlvMUr7n,utLxmPNeYKSf3T24OXskxB5 33-789-255-7342
slyly ironic packages detect furious accounts. ironic de
8877.820 Supplier#000007967 FRANCE
167966 Manufacturer#5 A3pi1BARM4nx6R,qrwFoRPU
16-442-147-9345 final deposits after the silent deposits ha
8862.240 Supplier#000003323 ROMANIA
73322 Manufacturer#3 W9 lYcsC9FwBqk3ltL
29-736-951-3710 unusual, pending theodolites integrate furiously slyly
even pinto beans. unusual sheaves sleep befor
8841.590 Supplier#000005750 ROMANIA
100729 Manufacturer#5
Erx3lAgu0g62iaHF9x50uMH4EgeN9hEG 29-344-502-5481
excuses after the blithely regular packages mold carefully deposits.
regular a
8781.710 Supplier#000003121 ROMANIA
13120 Manufacturer#5 wNqTogx238ZYCamFb,50v,bj
4IbNFW9Bvw1xP 29-707-291-5144 packages are quickly after the
final, even packages. furiously regular
8754.240 Supplier#000009407 UNITED KINGDOM
179406 Manufacturer#4 CHRCbkaWcf5B
33-903-970-9604 regular dependencies haggle across the carefully
bold
8691.060 Supplier#000004429 UNITED KINGDOM
126892 Manufacturer#2 k,BQms5UhoAF1B2Asi,fLib
33-964-337-5038 quickly special foxes against the furiously silent
platelets wake quickly after t
8655.990 Supplier#000006330 RUSSIA
193810 Manufacturer#2 UozlaENr0ytKe2w6CeIEWFWn
iO3S8Rae7Ou 32-561-198-3705 blithely even packages alongside
8638.360 Supplier#000002920 RUSSIA
75398 Manufacturer#1 Je2a8bszf3L 32-
122-621-7549 express deposits wake. furiously silent requests wake
carefully silent instru
8638.360 Supplier#000002920 RUSSIA
170402 Manufacturer#3 Je2a8bszf3L 32-
122-621-7549 express deposits wake. furiously silent requests wake
carefully silent instru
8607.690 Supplier#000006003 UNITED KINGDOM
76002 Manufacturer#2
EH9wADcEiuenM0NR08zDwMidw,52Y2RyILEiA 33-416-807-5206
always special foxes wake slyly bold, ironic accounts. ironic
instructions affix carefull
8569.520 Supplier#000005936 RUSSIA
5935 Manufacturer#5 jXaNZ6vwnEWJ2ksLZJpjtgt0bY2a3AU
32-644-251-7916 packages sleep furiously. special requests about the
fluffily even accounts detect
8564.120 Supplier#000000033 GERMANY
110032 Manufacturer#1
gfeKpYw3400L0SDywXA6Ya1Qmq1w6YB9f3R 17-138-897-9374
ironic instructions are. special pearls above
8553.820 Supplier#000003979 ROMANIA
143978 Manufacturer#4 BfmVhCAnCMY3jzpjUMy4CNWs9
HzpdQR7INJU 29-124-646-4897 express, ironic pinto beans cajole
around the express, even packages. qu
8517.230 Supplier#000009529 RUSSIA
37025 Manufacturer#5 e44R8o7JAIS9iMcr 32-
565-297-8775 furiously silent requests cajole furiously furiously ironic
foxes. slyly express p
8517.230 Supplier#000009529 RUSSIA
59528 Manufacturer#2 e44R8o7JAIS9iMcr 32-
565-297-8775 furiously silent requests cajole furiously furiously ironic
foxes. slyly express p
8503.700 Supplier#000006830 RUSSIA
44325 Manufacturer#4 BC4WFCYRUZyaIgchU 4S
32-147-878-5069 quickly regular excuses detect evenly around
8457.090 Supplier#000009456 UNITED KINGDOM
19455 Manufacturer#1 7SBhZs8gP1cJjT0Qf433YBk
33-858-440-4349 carefully final accounts sleep blithely special foxes.
slyly regular pinto beans alon
8441.400 Supplier#000003817 FRANCE
141302 Manufacturer#2 hU3fz3xL78 16-
339-356-5115 blithely blithe ideas are
8432.890 Supplier#000003990 RUSSIA
191470 Manufacturer#1
wehBBp1RQbfxAYDASS75MsywmsKHRVdkrvNe6m 32-839-509-
9301 final requests along the blithely ironic packages kindle against
the carefully fina
8431.400 Supplier#000002675 ROMANIA
5174 Manufacturer#1
HJFStOu9R5NGPOegKhgbzBdyvrG2yh8w 29-474-643-1443
express, final deposits cajole carefully. stealthily unusual requests
8407.040 Supplier#000005406 RUSSIA
162889 Manufacturer#4 j7 gYF5RW8DC5UrjKC
32-626-152-4621 quickly final sheaves boost. car
8386.080 Supplier#000008518 FRANCE
36014 Manufacturer#3
2jqzqqAVe9crMVGP,n9nTsQXulNLTUYoJjEDcqWV 16-618-780-7481
slyly ironic theodolites are slyly. dogged, pendin
8376.520 Supplier#000005306 UNITED KINGDOM
190267 Manufacturer#5 9t8Y8
QqSIsoADPt6NLdk,TP5zyRx41oBUlgoGc9 33-632-514-7931
furiously even instructions integrate during the furiously regular re
8348.740 Supplier#000008851 FRANCE
66344 Manufacturer#4 nWxi7GwEbjhw1 16-
796-240-2472 ironic instructions nag slyly against the slyly even
theodolites. requests alongside of
8338.580 Supplier#000007269 FRANCE
17268 Manufacturer#4 ZwhJSwABUoiB04,3
16-267-277-4365 ruthlessly regular asymptotes a
8328.460 Supplier#000001744 ROMANIA
69237 Manufacturer#5 oLo3fV64q2,FKHa3p,qHnS7Yzv,ps8
29-330-728-5873 blithely silent excuses are slyly above the furiously
even courts
8307.930 Supplier#000003142 GERMANY
18139 Manufacturer#1 dqblvV8dCNAorGlJ 17-
595-447-6026 theodolites sleep blithely carefully regular warhorses.
slyly regular ins

8231.610 Supplier#000009558 RUSSIA
192000 Manufacturer#2 mcdgen,yT1iJDHDS5fV
32-762-137-5858 slyly regular theodolites sleep fluffily express depos
8152.610 Supplier#000002731 ROMANIA
15227 Manufacturer#4 nluXJCuY1tu 29-
805-463-2030 gifts use. slyly silent ideas are carefully beneath the
silent instructions. slyly sil
8109.090 Supplier#000009186 FRANCE
99185 Manufacturer#1 wgfosrVPexl9pEXWywaqlBMDYYf
16-668-570-1402 quickly pending requests are blithely along the
ironic, final requests; instr
8102.620 Supplier#000003347 UNITED KINGDOM
18344 Manufacturer#5 m CtXS2S16i 33-
454-274-8532 packages grow special orbits. regular theodolites about
the carefully pe
8046.070 Supplier#000008780 FRANCE
191222 Manufacturer#3 AczzuE0UK9osj ,Lx0Jmh
16-473-215-6395 regular epitaphs integrate slyly.
8042.090 Supplier#000003245 RUSSIA
135705 Manufacturer#4
Dh8Ikg39onrbOL4DyTfGw8a9oKUX3d9Y 32-836-132-8872
carefully regular instructions integrate blithely silent foxes. furiously
express instructions haggl
8042.090 Supplier#000003245 RUSSIA
150729 Manufacturer#1
Dh8Ikg39onrbOL4DyTfGw8a9oKUX3d9Y 32-836-132-8872
carefully regular instructions integrate blithely silent foxes. furiously
express instructions haggl
7992.400 Supplier#000006108 FRANCE
118574 Manufacturer#1 8tBydnTDwUqfBfFV4l3
16-974-998-8937 regular pinto beans are after
7980.650 Supplier#000001288 FRANCE
13784 Manufacturer#4 zE,7HgVPrCn 16-
646-464-8247 unusual pinto beans cajole furiously according t
7950.370 Supplier#000008101 GERMANY
33094 Manufacturer#5 kkYvL6IuvojJgTNG IKkaXQDYgx8ILohj
17-627-663-8014 quickly regular requests are furiously. pending
deposits wake
7937.930 Supplier#000009012 ROMANIA
83995 Manufacturer#2 iUiTziH,Ek3i4lwSgunXMgrcTzwdb
29-250-925-9690 blithely bold ideas haggle quickly final, regular
request
7914.450 Supplier#000001013 RUSSIA
125988 Manufacturer#2 riRcntps4KEDtYScjpMIWeYF6mNnR
32-194-698-3365 final, ironic theodolites alongside of the ironic
7912.910 Supplier#000004211 GERMANY
159180 Manufacturer#5
2wQRVovHrm3,v03IKzfTd,1PYsFXQFFOG 17-266-947-7315
final requests integrate slyly above the silent, even
7912.910 Supplier#000004211 GERMANY
184210 Manufacturer#4
2wQRVovHrm3,v03IKzfTd,1PYsFXQFFOG 17-266-947-7315
final requests integrate slyly above the silent, even
7894.560 Supplier#000007981 GERMANY
85472 Manufacturer#4 NSJ96vMROAbeXP
17-963-404-3760 regular, even theodolites integrate carefully. bold,
special theodolites are slyly fluffily iron
7887.080 Supplier#000009792 GERMANY
164759 Manufacturer#3 Y28ITVeYriT3kIGdV2K8fSZ
V2UqT5H1Otz 17-988-938-4296 pending, ironic packages sleep
among the carefully ironic accounts. quickly final accounts
7871.500 Supplier#000007206 RUSSIA
104695 Manufacturer#1 3w fNCnrVmvJjE95sgWZzvW
32-432-452-7731 furiously dogged pinto beans cajole. bold, express
notornis until the slyly pending
7852.450 Supplier#000005864 RUSSIA
8363 Manufacturer#4 WCNfBPZeSXh3h,c 32-
454-883-3821 blithely regular deposits
7850.660 Supplier#000001518 UNITED KINGDOM
86501 Manufacturer#1 ONda3YJiHKJOC 33-
730-383-3892 furiously final accounts wake carefully idle requests.
even dolphins wake acc
7843.520 Supplier#000006683 FRANCE
11680 Manufacturer#4 2Z0JGkiv01Y00oCFwUGfviIbhzCdy
16-464-517-8943 carefully bold accounts doub

Number of rows retrieved is: 100

Query 3

-- 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

Query 4

-- 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

Query 5

-- 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

Query 6

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

Tag: Q6 Stream: -1 Sequence number: 5

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

```
REVENUE
---------------------
      123141078.228
```

Number of rows retrieved is: 1

Query 7

-- Query 07 - Var_0 Rev_01 - Volume Shipping Query

Tag: Q7 Stream: -1 Sequence number: 21

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

```
SUPP_NATION          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

Query 8

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

Tag: Q8  Stream: -1  Sequence number: 8

select
o_year,
sum(case
when nation = 'BRAZIL' then volume
else 0
end) / sum(volume) as mkt_share
from
(
select
year(o_orderdate) as o_year,
l_extendedprice * (1 - l_discount) as volume,
n2.n_name as nation
from
tpcd.part,
tpcd.supplier,
tpcd.lineitem,
tpcd.orders,
tpcd.customer,
tpcd.nation n1,
tpcd.nation n2,
tpcd.region
where
p_partkey = l_partkey
and s_suppkey = l_suppkey
and l_orderkey = o_orderkey
and o_custkey = c_custkey
and c_nationkey = n1.n_nationkey
and n1.n_regionkey = r_regionkey
and r_name = 'AMERICA'
and s_nationkey = n2.n_nationkey
and o_orderdate between date('1995-01-01') and date ('1996-12-31')
and p_type = 'ECONOMY ANODIZED STEEL'
) as all_nations
group by
o_year
order by
o_year
```

O_YEAR	MKT_SHARE
1995	0.034
1996	0.041

Number of rows retrieved is: 2

Query 9

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

Tag: Q9  Stream: -1  Sequence number: 3

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

NATION	O_YEAR	SUM_PROFIT
ALGERIA	1998	31342867.235
ALGERIA	1997	57138193.023
ALGERIA	1996	56140140.133
ALGERIA	1995	53051469.653
ALGERIA	1994	53867582.129
ALGERIA	1993	54942718.132
ALGERIA	1992	54628034.713
ARGENTINA	1998	30211185.708
ARGENTINA	1997	50805741.752
ARGENTINA	1996	51923746.575
ARGENTINA	1995	49298625.767
ARGENTINA	1994	50835610.109
ARGENTINA	1993	51646079.178
ARGENTINA	1992	50410314.995
BRAZIL	1998	27217924.383
BRAZIL	1997	48378669.199
BRAZIL	1996	50482870.357
BRAZIL	1995	47623383.635
BRAZIL	1994	47840165.726
BRAZIL	1993	49054694.035
BRAZIL	1992	48667639.084
CANADA	1998	30379833.769
CANADA	1997	50465052.311
CANADA	1996	52560501.390
CANADA	1995	52375332.809
CANADA	1994	52600364.659
CANADA	1993	52644504.073
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        47067209.332
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        50286902.852
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

[some lines deleted]
UNITED KINGDOM            1998          28494874.004
UNITED KINGDOM            1997          49381810.899
UNITED KINGDOM            1996          51386853.960
UNITED KINGDOM            1995          51509586.788
UNITED KINGDOM            1994          48086499.711
UNITED KINGDOM            1993          49166827.224
UNITED KINGDOM            1992          49349122.083
UNITED STATES             1998          25126238.946
UNITED STATES             1997          50077306.419
UNITED STATES             1996          48048649.470
UNITED STATES             1995          48809032.423
UNITED STATES             1994          49296747.183
UNITED STATES             1993          48029946.801
UNITED STATES             1992          48671944.498
VIETNAM                   1998        30442736.059
VIETNAM                   1997        50309179.794
VIETNAM                   1996        50488161.410
VIETNAM                   1995        49658284.612
VIETNAM                   1994        50596057.261
VIETNAM                   1993        50953919.152
VIETNAM                   1992        49613838.315


Number of rows retrieved is:    175
```

Query 10

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

Tag: Q10    Stream: -1    Sequence number: 18

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

C_CUSTKEY    C_NAME                  REVENUE
C_ACCTBAL          N_NAME                    C_ADDRESS
C_PHONE         C_COMMENT
-------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------------------------
----
     57040  Customer#000057040                734235.246
632.870  JAPAN                     Eioyzjf4pp                        22-895-
641-3466  requests sleep blithely about the furiously i
    143347  Customer#000143347                721002.695
2557.470  EGYPT                     1aReFYv,Kw4                       14-
742-935-3718  fluffily bold excuses haggle finally after the u
     60838  Customer#000060838                679127.308
2454.770  BRAZIL                    64EaJ5vMAHWJlBOxJklpNc2RJiWE
12-913-494-9813  furiously even pinto beans integrate under the
ruthless foxes; ironic, even dolphins across the slyl
    101998  Customer#000101998                637029.567
3790.890  UNITED KINGDOM            01c9CILnNtfOQYmZj
33-593-865-6378  accounts doze blithely! enticing, final deposits sleep
blithely special accounts. slyly express accounts pla
    125341  Customer#000125341                633508.086
4983.510  GERMANY                   S29ODD6bceU8QSuuEJznkNaK
17-582-695-5962  quickly express requests wake quickly blithely
     25501  Customer#000025501                620269.785
7725.040  ETHIOPIA
W556MXuoiaYCCZamJI,Rn0B4ACUGdkQ8DZ        15-874-808-6793
quickly special requests sleep evenly among the special deposits.
special deposi
    115831  Customer#000115831                596423.867
5098.100  FRANCE                    rFeBbEEyk dl
ne7zV5fDrmiq1oK09wV7pxqCgIc  16-715-386-3788  carefully bold
excuses sleep alongside of the thinly idle
     84223  Customer#000084223                594998.024
528.650  UNITED KINGDOM            nAVZCs6BaWap rrM27N
2qBnzc5WBauxbA        33-442-824-8191  pending, final ideas haggle
final requests. unusual, regular asymptotes affix according to the even
foxes.
     54289  Customer#000054289                585603.392
5583.020  IRAN                      vXCxoCsU0Bad5JQI ,oobkZ
20-834-292-4707  express requests sublate blithely regular requests.
regular, even ideas solve.
     39922  Customer#000039922                584878.113
7321.110  GERMANY
```

Zgy4s50l2GKN4pLDPBU8m342glw6R 17-147-757-8036 even pinto beans haggle. slyly bold accounts inte
6226 Customer#000006226 576783.761 2230.090 UNITED KINGDOM
8gPu8,NPGkfyQQ0hclYUGPIBWc,ybP5g, 33-657-701-3391 quickly final requests against the regular instructions wake blithely final instructions. pa
922 Customer#000000922 576767.533 3869.250 GERMANY
Az9RFaut7NkPnc5zSD2PwHgVwr4jRzq 17-945-916-9648 boldly final requests cajole blith
147946 Customer#000147946 576455.132 2030.130 ALGERIA iANyZHjqhyy7Ajah0pTrYyhJ 10-886-956-3143 furiously even accounts are blithely above the furiousl
115640 Customer#000115640 569341.193 6436.100 ARGENTINA Vtgfia9qI 7EpHgecU1X 11-411-543-4901 final instructions are slyly according to the
73606 Customer#000073606 568656.858 1785.670 JAPAN xuR0Tro5yChDfOCrjkd2ol 22-437-653-6966 furiously bold orbits about the furiously busy requests wake across the furiously quiet theodolites. d
110246 Customer#000110246 566842.982 7763.350 VIETNAM 7KzflgX MDOq7sOkI 31-943-426-9837 dolphins sleep blithely among the slyly final
142549 Customer#000142549 563537.237 5085.990 INDONESIA
ChqEoK43OysjdHbtKCp6dKqjNyvvi9 19-955-562-2398 regular, unusual dependencies boost slyly; ironic attainments nag fluffily into the unusual packages?
146149 Customer#000146149 557254.986 1791.550 ROMANIA s87fvzFQpU 29-744-164-6487 silent, unusual requests detect quickly slyly regul
52528 Customer#000052528 556397.351 551.790 ARGENTINA NFztyTOR10UOJ 11-208-192-3205 unusual requests detect. slyly dogged theodolites use slyly. deposit
23431 Customer#000023431 554269.536 3381.860 ROMANIA HgiV0phqhaIa9aydNoIlb 29-915-458-2654 instructions nag quickly. furiously bold accounts cajol

Number of rows retrieved is: 20

Query 11

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

Tag: Q11  Stream: -1  Sequence number: 15

select
ps_partkey,
sum(ps_supplycost * ps_availqty) as value
from
tpcd.partsupp,
tpcd.supplier,
tpcd.nation
where
ps_suppkey = s_suppkey
and s_nationkey = n_nationkey
and n_name = 'GERMANY'
group by
ps_partkey having
sum(ps_supplycost * ps_availqty) > (
select
sum(ps_supplycost * ps_availqty) * 0.00010000000
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
   141288      10837130.210
   152388      10830977.820
    33449      10830858.720
   149035      10826130.020
   162620      10814275.680
   118324      10791788.100
    38932      10777541.750
[ rows deleted ]

   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

Query 12

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

Tag: Q12    Stream: -1   Sequence number: 22

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

L_SHIPMODE  HIGH_LINE_COUNT  LOW_LINE_COUNT
----------------------------------------------
MAIL                6202           9324
```

```
SHIP                6200          9262


Number of rows retrieved is:      2
```

Query 13

```
-- Query 13 - Var_0 Rev_01 - Customer Distribution Query

Tag: Q13    Stream: -1   Sequence number: 10

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

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


Number of rows retrieved is:     42
```

Query 14

```
-- 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
```

Query 15

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

Tag: Q15a   Stream: -1   Sequence number: 16

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

```
S_SUPPKEY    S_NAME                    S_ADDRESS
S_PHONE         TOTAL_REVENUE
-------------------------------------------------------------------------------------------------
---------------------------
      8449  Supplier#000008449        Wp34zim9qYFbVctdW
20-469-856-8873          1772627.209


Number of rows retrieved is:      1
```

Query 16

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

Tag: Q16    Stream: -1   Sequence number: 13

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

P_BRAND    P_TYPE                    P_SIZE      SUPPLIER_CNT
----------------------------------------------------------------------
Brand#41   MEDIUM BRUSHED TIN              3          28
Brand#54   STANDARD BRUSHED COPPER         14          27
Brand#11   STANDARD BRUSHED TIN            23          24
Brand#11   STANDARD BURNISHED BRASS        36          24
Brand#15   MEDIUM ANODIZED NICKEL           3          24
Brand#15   SMALL ANODIZED BRASS            45          24
Brand#15   SMALL BURNISHED NICKEL          19          24
Brand#21   MEDIUM ANODIZED COPPER           3          24
Brand#22   SMALL BRUSHED NICKEL            3          24
Brand#22   SMALL BURNISHED BRASS           19          24
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
Brand#15   SMALL PLATED COPPER              49          20
Brand#15   STANDARD PLATED TIN               3          20
Brand#21   LARGE ANODIZED COPPER            36          20
Brand#21   LARGE BRUSHED TIN                3          20
Brand#21   MEDIUM ANODIZED COPPER           14          20
Brand#21   PROMO BRUSHED TIN                36          20
Brand#21   PROMO POLISHED NICKEL            45          20
Brand#21   SMALL ANODIZED COPPER             9          20
Brand#21   SMALL POLISHED NICKEL            23          20
Brand#22   LARGE ANODIZED COPPER            36          20
Brand#22   LARGE BRUSHED COPPER              49          20
Brand#22   PROMO ANODIZED TIN               49          20
Brand#22   PROMO POLISHED BRASS             45          20
Brand#22   SMALL BURNISHED STEEL            45          20
Brand#23   MEDIUM ANODIZED STEEL            45          20
Brand#23   PROMO POLISHED STEEL             23          20
Brand#23   STANDARD BRUSHED TIN             14          20
Brand#23   STANDARD PLATED NICKEL           36          20
Brand#24   PROMO PLATED COPPER              49          20
Brand#24   PROMO PLATED STEEL               49          20
Brand#24   PROMO POLISHED STEEL              9          20
Brand#24   STANDARD BRUSHED TIN             36          20
Brand#25   LARGE ANODIZED BRASS              3          20
Brand#25   PROMO BURNISHED TIN               3          20
Brand#31   ECONOMY POLISHED NICKEL           3          20
Brand#31   MEDIUM PLATED TIN                45          20
Brand#31   SMALL ANODIZED STEEL             14          20
Brand#32   ECONOMY ANODIZED COPPER          36          20
Brand#32   ECONOMY BRUSHED NICKEL           49          20
Brand#32   LARGE ANODIZED TIN               19          20
Brand#32   MEDIUM BURNISHED COPPER           19          20
Brand#32   SMALL ANODIZED STEEL             45          20
Brand#33   ECONOMY POLISHED COPPER           19          20
Brand#33   PROMO PLATED NICKEL              14          20
Brand#33   SMALL POLISHED TIN                9          20
Brand#33   STANDARD ANODIZED BRASS           49          20
Brand#33   STANDARD BURNISHED BRASS          45          20
Brand#34   ECONOMY BRUSHED NICKEL            49          20
Brand#34   LARGE BRUSHED BRASS               19          20
Brand#34   SMALL BRUSHED TIN                 3          20
```

```
Brand#34    STANDARD PLATED COPPER          9          20
Brand#35    LARGE ANODIZED NICKEL           3          20
Brand#35    MEDIUM ANODIZED BRASS           45         20
Brand#35    MEDIUM ANODIZED STEEL           23         20
Brand#35    PROMO ANODIZED COPPER           49         20
Brand#35    SMALL POLISHED COPPER           14         20
Brand#41    LARGE ANODIZED STEEL            3          20
Brand#41    LARGE BRUSHED NICKEL            23         20
Brand#41    LARGE BURNISHED COPPER          3          20
Brand#41    MEDIUM PLATED STEEL             19         20
Brand#41    SMALL BURNISHED COPPER          23         20
Brand#42    MEDIUM BURNISHED BRASS          14         20
Brand#42    SMALL BURNISHED COPPER          3          20
Brand#43    ECONOMY POLISHED COPPER         9          20
Brand#43    SMALL PLATED STEEL              3          20
Brand#43    STANDARD BURNISHED TIN          23         20
Brand#44    LARGE ANODIZED STEEL            23         20

[ rows deleted ]

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

Query 17

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

Tag: Q17 Stream: -1 Sequence number: 6

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

```
AVG_YEARLY
----------
    348406.054
```

Number of rows retrieved is: 1

Query 18

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

Tag: Q18 Stream: -1 Sequence number: 7

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

```
C_NAME                  C_CUSTKEY    O_ORDERKEY
O_ORDERDATE  O_TOTALPRICE         6
-------------------------------------------------------------------------------------------------
----------------------
Customer#000128120           128120           4722021  1994-04-07
544089.090           323.000
Customer#000144617           144617           3043270  1997-02-12
530604.440           317.000
Customer#000013940            13940           2232932  1997-04-13
522720.610           304.000
Customer#000066790            66790           2199712  1996-09-30
515531.820           327.000
Customer#000046435            46435           4745607  1997-07-03
508047.990           309.000
Customer#000015272            15272           3883783  1993-07-28
500241.330           302.000
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
```

Query 19

```
-- Query 19 - Var_0 Rev_01 - Discounted Revenue Query

Tag: Q19    Stream: -1   Sequence number: 19

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

```
and l_shipinstruct = 'DELIVER IN PERSON'
)
or
(
p_partkey = l_partkey
and p_brand = 'Brand#23'
and p_container in ('MED BAG', 'MED BOX', 'MED PKG', 'MED PACK')
and l_quantity >= 10 and l_quantity <= 10 + 10
and p_size between 1 and 10
and l_shipmode in ('AIR', 'AIR REG')
and l_shipinstruct = 'DELIVER IN PERSON'
)
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
```

Query 20

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

Tag: Q20    Stream: -1   Sequence number: 4

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

S_NAME                    S_ADDRESS
[ rows deleted from output below ]
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        Q9C4rfJ26oijVPqqcqVXeRl
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        nLKHUOn2MI9TOA06Znq9GEMclIMO2
Supplier#000002571        JZUugz04c iJFLrlGsz9O N,W 1rVHNIReyq
Supplier#000002585        CsPoKpw2QuTY4AV1NkWuttneIa4SN
Supplier#000002630        ZlQAvjNUY9KH5ive zm7k VlPiDl7CCo21
Supplier#000002719        4nnzQI2CbqREQUuIsXTBVUkaP4mNS3
Supplier#000002721        HVdFAN2JHMQSpKm
Supplier#000002730        lIFxR4fzm31C6,muzJwl84z
Supplier#000002775        yDclaDaBD4ihH
Supplier#000002853        rTNAOltXka
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 VClZqMSZVa r g7LTdcg29GbTE7rl1x
Supplier#000003095 HxON3jJhUi3zjt,r mTD
Supplier#000003201 E87yws6l,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 zvFJlzS,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 yvSsKNSTL5HLXBET4IuOsPNLxKzAMk
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 OfvYPs3lo,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 TaIC5m0pDrO6DZbngfmGmqe
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 4sSDA4ACRekINjEm5T6b
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
```

Query 21

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

Tag: Q21    Stream: -1   Sequence number: 9

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

S_NAME                    NUMWAIT
----------------------------------------
Supplier#000002829              20
Supplier#000005808              18
Supplier#000000262              17
Supplier#000000496              17
Supplier#000002160              17
Supplier#000002301              17
Supplier#000002540              17
Supplier#000003063              17
Supplier#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
```

Query 22

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

Tag: Q22    Stream: -1    Sequence number: 12

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

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


Number of rows retrieved is:      7
```

C.2 First 10 rows of Test Database Tables

```
SELECT * FROM TPCD.REGION FETCH FIRST 10 ROWS ONLY

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

  5 record(s) selected.


SELECT * FROM TPCD.NATION FETCH FIRST 10 ROWS ONLY

N_NATIONKEY N_NAME                  N_REGIONKEY N_COMMENT
----------- ----------------------- ----------- ------------------------------------------------
------------------------------------------------------------------------------------------------
------------
          0 ALGERIA                           0 final accounts wake quickly.
special reques
          5 ETHIOPIA                          0 fluffily ruthless requests integrate
fluffily. pending ideas wake blithely acco
         14 KENYA                             0 ironic requests boost. quickly
pending pinto beans cajole slyly slyly even deposits. ironic packages
         15 MOROCCO                           0 ideas according to the fluffily
final pinto beans sleep furiously
         16 MOZAMBIQUE                        0 ironic courts wake fluffily
even, bold deposi
```

```
         1 ARGENTINA                    1 idly final instructions cajole
stealthily. regular instructions wake carefully blithely express accounts.
fluffi
         2 BRAZIL                       1 always pending pinto beans sleep
sil
         3 CANADA                       1 foxes among the bold requests
        17 PERU                         1 final, final accounts sleep slyly
across the requests.
        24 UNITED STATES                1 blithely regular deposits
serve furiously blithely regular warthogs! slyly fi

  10 record(s) selected.


SELECT * FROM TPCD.PART FETCH FIRST 10 ROWS ONLY

P_PARTKEY   P_NAME                                        P_MFGR
P_BRAND    P_TYPE                P_SIZE      P_CONTAINER
P_RETAILPRICE          P_COMMENT
----------- ---------------------------------------------- ------------------------- -
--------- ------------------------- ----------- ----------- ------------------------ -----------
-----------
         43 medium khaki chocolate rosy blush
Manufacturer#4           Brand#44   PROMO POLISHED STEEL
5 WRAP CASE     +9.43040000000000E+002 carefully iro
         98 frosted goldenrod chartreuse dark honeydew
Manufacturer#5           Brand#54   STANDARD ANODIZED BRASS
22 MED JAR       +9.98090000000000E+002 furiou
        144 wheat brown orange almond aquamarine
Manufacturer#1           Brand#14   SMALL ANODIZED TIN
26 SM BOX        +1.04414000000000E+003 blithely bold r
        232 ivory purple spring tan cornsilk                    Manufacturer#5
Brand#53   LARGE BURNISHED NICKEL           50 SM PKG
+1.13223000000000E+003 quick deposits enga
        242 magenta deep lawn linen navy
Manufacturer#3           Brand#35   SMALL POLISHED STEEL
42 LG BAG        +1.14224000000000E+003 final pearls wake b
        474 gainsboro chiffon dodger orchid royal
Manufacturer#1           Brand#14   ECONOMY PLATED STEEL
45 LG PACK       +1.37447000000000E+003 fluffi
        612 midnight deep misty magenta honeydew
Manufacturer#4           Brand#42   PROMO PLATED STEEL
19 LG BOX        +1.51261000000000E+003 ironic, expr
        643 beige navy dim green forest
Manufacturer#3           Brand#32   MEDIUM POLISHED STEEL
8 MED DRUM       +1.54364000000000E+003 furiously regular requ
        647 bisque violet dim lawn drab                         Manufacturer#3
Brand#35   LARGE BURNISHED STEEL           38 MED PKG
+1.54764000000000E+003 quickly spec
        659 ivory green pink orange chartreuse
Manufacturer#3           Brand#34   MEDIUM BRUSHED BRASS
20 LG JAR        +1.55965000000000E+003 even,

  10 record(s) selected.


SELECT * FROM TPCD.SUPPLIER FETCH FIRST 10 ROWS ONLY

S_SUPPKEY   S_NAME                 S_ADDRESS
S_NATIONKEY S_PHONE         S_ACCTBAL              S_COMMENT
----------- ------------------------ ---------------------------------------- ---------- -----
---------- ------------------------ ---------------------------------------------------------
-----------------------------------------
         474 Supplier#000000474         USHBMdX8iFodU
0 10-327-319-7717   +5.22621000000000E+003 pending, express
courts along the carefully express accounts use quickly final instr
       12278 Supplier#000012278         SRh W8prdPpDwkNTtc5
0 10-449-254-7798   +5.00695000000000E+003 pending, even
accounts boost carefully according to the foxes. slyly slow
dependencie
       13694 Supplier#000013694         nIR4UPXyoDgbf26ysfvQS
0 10-412-354-1216   +1.66888000000000E+003 furiously express
requests affix fluffily
       15501 Supplier#000015501         PMmxCusN Dfqh
0 10-290-188-3714   +6.90513000000000E+003 instructions after the
furiously ironic foxes nag carefully among the pending re
       19748 Supplier#000019748
,mqah0n6g5v80R6lcLsfdWOO7msX8SX,PTdt              0 10-780-820-
6322   -5.16650000000000E+002 ironic ideas are boldly regular
instructions. blithely final acco
       20567 Supplier#000020567         q9x2SaQFeR41Ih6J7J4j
xEGXQFFAFookD95nb0            0 10-225-379-5566
+9.95998000000000E+003 carefully regular platelets sleep atop the
theodolites. bold accounts caj
       20894 Supplier#000020894         F6Yeyf,rg5l0SAv15Pff
0 10-589-924-4741   +5.70489000000000E+003 fluffily regular dugouts
n
       25091 Supplier#000025091         fZM af
a,0TtKKbDJaqBAinSxrMc5igb5tjnWbV            0 10-157-662-6929
+7.42398000000000E+003 carefully express pinto beans nag slyly.
furiously fin
       30087 Supplier#000030087         y2Tko2qt8iPxryGJPb
0 10-875-191-5633   +6.39575000000000E+003 furiously regular
requests acro
       43051 Supplier#000043051         eK1KoGMjMApto0TR,dnH
0 10-287-874-1814   +2.12459000000000E+003 warhorses affix
furiously unusual s

  10 record(s) selected.


SELECT * FROM TPCD.PARTSUPP FETCH FIRST 10 ROWS ONLY

PS_PARTKEY  PS_SUPPKEY  PS_AVAILQTY PS_SUPPLYCOST
PS_COMMENT
----------- ----------- ----------- ------------------------ ---------------------------------
---------------------------------------------------------------------------------------------
---------------------------------------------------------------------
         43          44        3211   +8.05780000000000E+002 final, express
dependencies sleep according to the express requests. bold, regular
accounts detect outside the slyly
         43    25000044        6770   +4.93190000000000E+002 furiously
special pinto beans cajole. ironic decoys across the
         43    50000044        9506   +4.93650000000000E+002 carefully
fluffy accounts across the blithely final accounts hang slyly according to
the furiously special platelets. sil
         43    75000044        3232   +3.07120000000000E+002 bold
packages wake blithely above the furiously bold
         98          99        9486   +9.08210000000000E+002 deposits
haggle busily express deposits. furiously blithe platelets
         98    25000099        8550   +6.57160000000000E+002 express,
final deposits haggle along the regular foxes. carefully regular excuses
wake against the carefully even pinto beans. furiously express pinto
beans
         98    50000099        3443   +1.39000000000000E+002 express,
express pinto beans wake blithely. silent, pending requests around the
special packages cajole after the quietly regular accounts. somas
sleep.
         98    75000099        3759   +8.11550000000000E+002 blithely
silent instructions promise furiously across the blithely regular
dependencies. unusual packages print across the ironic pinto beans.
orbits sleep blithely against t
        144         145        6295   +4.57370000000000E+002 carefully
fluffy deposits wake slyly at the furiously final packages. regular
instructions nag sometimes even dolphins. bold packages across the
requests use unusual requests. qu
        144    25000145         494   +8.49960000000000E+002 quickly
silent accounts will detect quickly across the doggedly express
deposits. quick p
```

```
  10 record(s) selected.


SELECT * FROM TPCD.CUSTOMER FETCH FIRST 10 ROWS ONLY

C_CUSTKEY   C_NAME                  C_ADDRESS
C_NATIONKEY C_PHONE        C_ACCTBAL
C_MKTSEGMENT C_COMMENT
----------- ------------------------ ---------------------------------------- ----------- -----
--------- --------------- ------------ ----------------------------------------------------
----------------------------------------------------------------------------
          43 Customer#0000000043
ouSbjHk8lh5fKX3zGso3ZSIj9Aa3PoaFd                19 29-316-665-
2897   +9.90428000000000E+003 MACHINERY    idly regular
sentiments affix. slyly pending foxes around the deposits haggle
according to th
          98 Customer#0000000098       7yiheXNSpuEAwbswDW
12 22-885-845-6889   -5.51370000000000E+002 BUILDING     blithely
final foxes along the unusual realms detect accounts. idle theodolites
use pinto beans. ev
         144 Customer#0000000144
VxYZ3ebhgbltnetaGjNC8qCccjYU05 fePLOno8y           1 11-717-379-
4478   +6.41731000000000E+003 MACHINERY    fluffily regular
requests at the blithely even theodolites haggle around the requests.
special frets x-
         232 Customer#0000000232
oA9o,3YOXu2rzKONdd,cxpqCFXUv5kuxBYKp              22 32-283-563-
2674   +5.54710000000000E+002 HOUSEHOLD    ideas eat furiously
carefully special instructions
         242 Customer#0000000242
apgzK3HWAjKHFteJ16Tg3OERViesqBbx                  3 13-324-350-
3564   +1.97541000000000E+003 MACHINERY    carefully express
ideas nag final requests. slyly daring accounts cajole slyly blithely pend
         474 Customer#0000000474       mvEKw,6zT0V8Yb2yTG
hu990UX                     21 31-247-536-6143
+9.16547000000000E+003 MACHINERY    packages after the
carefully ironic packages haggle blithely against the carefully even
foxes. regular deposits ab
         612 Customer#0000000612       oNFqorGhq3a3woEp5q8xVDX
14 24-818-339-9984   +7.66916000000000E+003 HOUSEHOLD
final, final ideas mold after the quickly special dependencies.
escapades sleep slyly. e
         643 Customer#0000000643       9T 2avhfyF PQ
0 10-978-597-2747   +5.18470000000000E+003 FURNITURE
quickly even instructions sleep slyly around the furiously special
instructions. quickly silent deposits integrate c
         647 Customer#0000000647
2Bx7,7i87h5cagC,ZBz49lyiziLqQoD                   1 11-873-931-2886   -
1.32970000000000E+002 BUILDING     ironic accounts are furiously!
regular instruc
         659 Customer#0000000659       ThR9miOedPuwVEZyz
3MMjHPwB                    0 10-834-287-1466
+5.29768000000000E+003 HOUSEHOLD    final requests integrate
carefully above the carefully ironic foxes. furiously bold re

  10 record(s) selected.


SELECT * FROM TPCD.ORDERS FETCH FIRST 10 ROWS ONLY

O_ORDERKEY          O_CUSTKEY   O_ORDERSTATUS
O_TOTALPRICE          O_ORDERDATE O_ORDERPRIORITY
O_CLERK         O_SHIPPRIORITY O_COMMENT
------------------- ----------- ------------ ---------------------- ----------- ------------
--- --------------- ------------- ----------------------------------------------------------
----------------------------------------
          1719271   463445426 F             +6.51015800000000E+004
01/01/1992  1-URGENT       Clerk#009498513             0 blithely bold
packages haggle furiously pend
          4813991   470080876 F             +1.16941000000000E+004
01/01/1992  5-LOW          Clerk#006220763             0 blithely pending
ideas sleep furiously pending deposits. furiously
          7935557   638621302 F             +1.90003310000000E+005
01/01/1992  2-HIGH         Clerk#002965924             0 regular
accounts sleep
          8508070   816415318 F             +1.89338530000000E+005
01/01/1992  4-NOT SPECIFIED Clerk#005026238             0 furiously
express ideas hang q
          9126724  1414635590 F             +2.80025650000000E+005
01/01/1992  4-NOT SPECIFIED Clerk#009148215             0 regular
foxes haggle a
         10198884  1336517476 F             +4.57434000000000E+005
01/01/1992  4-NOT SPECIFIED Clerk#008305642             0 regular
foxes after the unusual packages wake furi
         10451045   964130441 F             +1.81500090000000E+005
01/01/1992  5-LOW          Clerk#000290660             0 slyly bold pinto
beans shall have to cajole regular,
         10883684   940194166 F             +8.45918900000000E+004
01/01/1992  1-URGENT       Clerk#005847571             0 fluffily final
theodolites wake furiously. ideas maintain carefully fu
         12518016  1104568180 F             +1.80970020000000E+005
01/01/1992  3-MEDIUM       Clerk#009764636             0 blithely even
dependencies nod slyly final pinto beans. ironic, fina
         13642599  1480675220 F             +2.26428820000000E+005
01/01/1992  1-URGENT       Clerk#005683954             0 furiously
unusual deposits beside

  10 record(s) selected.


SELECT * FROM TPCD.LINEITEM FETCH FIRST 10 ROWS ONLY

L_ORDERKEY          L_PARTKEY   L_SUPPKEY   L_LINENUMBER
L_QUANTITY            L_EXTENDEDPRICE         L_DISCOUNT
L_TAX                 L_RETURNFLAG L_LINESTATUS L_SHIPDATE
L_COMMITDATE L_RECEIPTDATE L_SHIPINSTRUCT
L_SHIPMODE L_COMMENT
------------------- ----------- ----------- ------------ ----------------------- ------------
---------- ---------- ----------------------- ----------------------- ------------ ------------
-- ------------ -------------- ------------------------- ---------- ----------------------------
---------------
          64987014  1055286870    30286901           6
+4.70000000000000E+001   +8.47931700000000E+004
+4.00000000000000E-002   +0.00000000000000E+000 R           F
01/02/1992 03/02/1992   01/26/1992    TAKE BACK RETURN
REG AIR    blithely ironic deposits alongside o
         108527269  1360219196    10219223           1
+4.20000000000000E+001   +5.23819800000000E+004
+6.00000000000000E-002   +6.00000000000000E-002 R           F
01/02/1992 03/16/1992   01/19/1992    DELIVER IN PERSON
FOB        quickly iro
         118117255   524293045    74293056           1
+4.60000000000000E+001   +4.65441800000000E+004
+8.00000000000000E-002   +4.00000000000000E-002 R           F
01/02/1992 02/05/1992   01/18/1992    NONE                RAIL
pending ideas about t
         192768546   489415143    89415144           2
+4.10000000000000E+001   +5.05808800000000E+004
+5.00000000000000E-002   +5.00000000000000E-002 A           F
01/02/1992 03/19/1992   01/07/1992    TAKE BACK RETURN
MAIL       slyly pending packa
         273842695  1857012865    57012866           2
+1.00000000000000E+001   +1.88502000000000E+004
+0.00000000000000E+000   +1.00000000000000E-002 R           F
01/02/1992 03/05/1992   01/26/1992    COLLECT COD               FOB
fluffily pending platelets accord
         308431303  1475460844      460859           2
+1.50000000000000E+001   +2.89662000000000E+004
+7.00000000000000E-002   +6.00000000000000E-002 R           F
```

```
01/02/1992 02/26/1992   01/04/1992    NONE                 SHIP
slyly regular deposits hag
        375678951   113992557    38992559          2
+2.40000000000000E+001   +3.94526400000000E+004
+5.00000000000000E-002   +5.00000000000000E-002 A          F
01/02/1992 03/03/1992   01/29/1992    DELIVER IN PERSON
RAIL       unusual, final theodolites affix carefull
        389990370   414065632    39065637          3
+4.00000000000000E+000   +6.30772000000000E+003
+4.00000000000000E-002   +1.00000000000000E-002 R          F
01/02/1992 03/28/1992   01/10/1992    TAKE BACK RETURN
AIR        even packages are after the deposits. bl
        505643460   384698450    84698451          5
+6.00000000000000E+000   +8.57532000000000E+003
+8.00000000000000E-002   +4.00000000000000E-002 R          F
01/02/1992 02/10/1992   01/27/1992    TAKE BACK RETURN
REG AIR    even, final ideas s
        550087042   204687669     4687670          5
+1.80000000000000E+001   +2.96357400000000E+004
+1.00000000000000E-001   +3.00000000000000E-002 A          F
01/02/1992 02/02/1992   01/16/1992    TAKE BACK RETURN
SHIP       final, unusual accoun

 10 record(s) selected.
```

C.3 Query Substitution Parameters

```
Power stream    Seed = 511093318
-- TPC TPC-H Parameter Substitution (Version 2.1.1b)
-- using 511093318 as a seed to the RNG
Q1  DELTA       95
Q2  SIZE        11
   TYPE        STEEL
   REGION      AFRICA
Q3  SEGMENT     HOUSEHOLD
   DATE        1995-03-03
Q4  DATE        1993-07-01
Q5  REGION      AFRICA
   DATE        1997-01-01
Q6  DATE        1997-01-01
   DISCOUNT    0.06
   QUANTITY    25
Q7  NATION1     MOROCCO
   NATION2     UNITED STATES
Q8  NATION      UNITED STATES
   REGION      AMERICA
   TYPE        MEDIUM BRUSHED COPPER
Q9  COLOR       lime
Q10 DATE        1993-07-01
Q11 NATION      ALGERIA
   FRACTION    0.0000000100
Q12 SHIPMODE1   SHIP
   SHIPMODE2   MAIL
   DATE        1995-01-01
Q13 WORD1       unusual
   WORD2       requests
Q14 DATE        1995-04-01
Q15 DATE        1994-10-01
Q16 BRAND       Brand#33
   TYPE        STANDARD POLISHED
   SIZE1       11
   SIZE2       19
   SIZE3       14
   SIZE4       13
   SIZE5       27
   SIZE6       10
   SIZE7       15
   SIZE8       12
Q17 BRAND       Brand#31
   CONTAINER   WRAP CAN
Q18 QUANTITY    312
Q19 BRAND1      Brand#55
   BRAND2      Brand#24
   BRAND3      Brand#24
   QUANTITY1   3
   QUANTITY2   11
   QUANTITY3   21
Q20 COLOUR      steel
   DATE        1994-01-01
   NATION      MOROCCO
Q21 NATION      FRANCE
Q22 I1          30
   I2          19
   I3          33
   I4          15
   I5          17
   I6          14
   I7          24

Throughput Stream = 1    Seed = 511093319
-- TPC TPC-H Parameter Substitution (Version 2.1.1b)
-- using 511093319 as a seed to the RNG
Q1  DELTA       103
Q2  SIZE        49
   TYPE        BRASS
   REGION      EUROPE
Q3  SEGMENT     AUTOMOBILE
   DATE        1995-03-20
Q4  DATE        1996-02-01
Q5  REGION      AMERICA
   DATE        1993-01-01
Q6  DATE        1993-01-01
   DISCOUNT    0.04
   QUANTITY    25
Q7  NATION1     GERMANY
   NATION2     MOZAMBIQUE
Q8  NATION      MOZAMBIQUE
   REGION      AFRICA
   TYPE        MEDIUM PLATED COPPER
Q9  COLOR       khaki
Q10 DATE        1994-05-01
Q11 NATION      JORDAN
   FRACTION    0.0000000100
Q12 SHIPMODE1   FOB
   SHIPMODE2   TRUCK
   DATE        1997-01-01
Q13 WORD1       unusual
   WORD2       requests
Q14 DATE        1995-07-01
Q15 DATE        1997-05-01
Q16 BRAND       Brand#23
   TYPE        LARGE ANODIZED
   SIZE1       2
   SIZE2       24
   SIZE3       1
   SIZE4       11
   SIZE5       6
   SIZE6       25
   SIZE7       41
   SIZE8       10
Q17 BRAND       Brand#33
   CONTAINER   SM CASE
Q18 QUANTITY    314
Q19 BRAND1      Brand#12
   BRAND2      Brand#52
   BRAND3      Brand#23
   QUANTITY1   9
   QUANTITY2   12
   QUANTITY3   28
Q20 COLOUR      frosted
```

```
   DATE        1997-01-01
   NATION      ETHIOPIA
Q21 NATION      UNITED KINGDOM
Q22 I1          15
   I2          23
   I3          27
   I4          30
   I5          33
   I6          29
   I7          34

Throughput Stream = 2    Seed = 511093320
-- TPC TPC-H Parameter Substitution (Version 2.1.1b)
-- using 511093320 as a seed to the RNG
Q1  DELTA       111
Q2  SIZE        37
   TYPE        NICKEL
   REGION      AFRICA
Q3  SEGMENT     HOUSEHOLD
   DATE        1995-03-05
Q4  DATE        1993-11-01
Q5  REGION      ASIA
   DATE        1993-01-01
Q6  DATE        1993-01-01
   DISCOUNT    0.09
   QUANTITY    24
Q7  NATION1     UNITED STATES
   NATION2     INDIA
Q8  NATION      INDIA
   REGION      ASIA
   TYPE        MEDIUM BURNISHED TIN
Q9  COLOR       green
Q10 DATE        1993-02-01
Q11 NATION      ARGENTINA
   FRACTION    0.0000000100
Q12 SHIPMODE1   TRUCK
   SHIPMODE2   FOB
   DATE        1995-01-01
Q13 WORD1       unusual
   WORD2       accounts
Q14 DATE        1995-10-01
Q15 DATE        1995-02-01
Q16 BRAND       Brand#53
   TYPE        PROMO BURNISHED
   SIZE1       5
   SIZE2       35
   SIZE3       33
   SIZE4       41
   SIZE5       50
   SIZE6       17
   SIZE7       6
   SIZE8       10
Q17 BRAND       Brand#35
   CONTAINER   SM JAR
Q18 QUANTITY    315
Q19 BRAND1      Brand#14
   BRAND2      Brand#45
   BRAND3      Brand#12
   QUANTITY1   4
   QUANTITY2   13
   QUANTITY3   24
Q20 COLOUR      puff
   DATE        1996-01-01
   NATION      SAUDI ARABIA
Q21 NATION      MOZAMBIQUE
Q22 I1          24
   I2          13
   I3          34
   I4          27
   I5          22
   I6          33
   I7          20

Throughput Stream = 3    Seed = 511093321
-- TPC TPC-H Parameter Substitution (Version 2.1.1b)
-- using 511093321 as a seed to the RNG
Q1  DELTA       119
Q2  SIZE        25
   TYPE        TIN
   REGION      EUROPE
Q3  SEGMENT     AUTOMOBILE
   DATE        1995-03-22
Q4  DATE        1996-05-01
Q5  REGION      EUROPE
   DATE        1993-01-01
Q6  DATE        1993-01-01
   DISCOUNT    0.07
   QUANTITY    24
Q7  NATION1     MOZAMBIQUE
   NATION2     ALGERIA
Q8  NATION      ALGERIA
   REGION      AFRICA
   TYPE        SMALL BRUSHED TIN
Q9  COLOR       floral
Q10 DATE        1993-11-01
Q11 NATION      KENYA
   FRACTION    0.0000000100
Q12 SHIPMODE1   RAIL
   SHIPMODE2   FOB
   DATE        1996-01-01
Q13 WORD1       express
   WORD2       accounts
Q14 DATE        1996-02-01
Q15 DATE        1997-08-01
Q16 BRAND       Brand#33
   TYPE        SMALL POLISHED
   SIZE1       38
   SIZE2       50
   SIZE3       17
   SIZE4       34
   SIZE5       20
   SIZE6       9
   SIZE7       14
   SIZE8       40
Q17 BRAND       Brand#32
   CONTAINER   SM CAN
Q18 QUANTITY    313
Q19 BRAND1      Brand#11
   BRAND2      Brand#22
   BRAND3      Brand#11
   QUANTITY1   9
   QUANTITY2   14
   QUANTITY3   21
Q20 COLOUR      chiffon
   DATE        1994-01-01
   NATION      INDONESIA
Q21 NATION      INDIA
Q22 I1          19
   I2          32
   I3          15
   I4          18
   I5          29
   I6          14
   I7          30

Throughput Stream = 4    Seed = 511093322
-- TPC TPC-H Parameter Substitution (Version 2.1.1b)
-- using 511093322 as a seed to the RNG
Q1  DELTA       66
Q2  SIZE        13
```

```
   TYPE        COPPER
   REGION      AMERICA
Q3  SEGMENT     FURNITURE
   DATE        1995-03-07
Q4  DATE        1994-02-01
Q5  REGION      MIDDLE EAST
   DATE        1993-01-01
Q6  DATE        1993-01-01
   DISCOUNT    0.04
   QUANTITY    25
Q7  NATION1     INDIA
   NATION2     PERU
Q8  NATION      PERU
   REGION      AMERICA
   TYPE        SMALL PLATED TIN
Q9  COLOR       dark
Q10 DATE        1994-08-01
Q11 NATION      BRAZIL
   FRACTION    0.0000000100
Q12 SHIPMODE1   AIR
   SHIPMODE2   FOB
   DATE        1996-01-01
Q13 WORD1       express
   WORD2       accounts
Q14 DATE        1996-05-01
Q15 DATE        1995-05-01
Q16 BRAND       Brand#23
   TYPE        LARGE BRUSHED
   SIZE1       5
   SIZE2       17
   SIZE3       40
   SIZE4       31
   SIZE5       3
   SIZE6       29
   SIZE7       10
   SIZE8       27
Q17 BRAND       Brand#34
   CONTAINER   LG CASE
Q18 QUANTITY    314
Q19 BRAND1      Brand#23
   BRAND2      Brand#15
   BRAND3      Brand#11
   QUANTITY1   4
   QUANTITY2   15
   QUANTITY3   28
Q20 COLOUR      mint
   DATE        1993-01-01
   NATION      UNITED STATES
Q21 NATION      ALGERIA
Q22 I1          14
   I2          25
   I3          11
   I4          13
   I5          23
   I6          29
   I7          17

Throughput Stream = 5   Seed = 511093323
-- TPC TPC-H Parameter Substitution (Version 2.1.1b)
-- using 511093323 as a seed to the RNG
Q1  DELTA       74
Q2  SIZE        50
   TYPE        BRASS
   REGION      EUROPE
Q3  SEGMENT     AUTOMOBILE
   DATE        1995-03-24
Q4  DATE        1996-09-01
Q5  REGION      AFRICA
   DATE        1994-01-01
Q6  DATE        1994-01-01
   DISCOUNT    0.02
   QUANTITY    24
Q7  NATION1     ALGERIA
   NATION2     INDONESIA
Q8  NATION      INDONESIA
   REGION      ASIA
   TYPE        SMALL ANODIZED TIN
Q9  COLOR       chocolate
Q10 DATE        1993-05-01
Q11 NATION      MOROCCO
   FRACTION    0.0000000100
Q12 SHIPMODE1   REG AIR
   SHIPMODE2   FOB
   DATE        1996-01-01
Q13 WORD1       express
   WORD2       accounts
Q14 DATE        1996-08-01
Q15 DATE        1993-02-01
Q16 BRAND       Brand#53
   TYPE        STANDARD BURNISHED
   SIZE1       15
   SIZE2       1
   SIZE3       12
   SIZE4       29
   SIZE5       24
   SIZE6       3
   SIZE7       42
   SIZE8       31
Q17 BRAND       Brand#31
   CONTAINER   LG JAR
Q18 QUANTITY    312
Q19 BRAND1      Brand#25
   BRAND2      Brand#43
   BRAND3      Brand#55
   QUANTITY1   10
   QUANTITY2   16
   QUANTITY3   24
Q20 COLOUR      yellow
   DATE        1996-01-01
   NATION      KENYA
Q21 NATION      PERU
Q22 I1          21
   I2          33
   I3          31
   I4          11
   I5          25
   I6          20
   I7          27
Throughput Stream = 6   Seed = 511093324
-- TPC TPC-H Parameter Substitution (Version 2.1.1b)
-- using 511093324 as a seed to the RNG
Q1  DELTA       82
Q2  SIZE        38
   TYPE        NICKEL
   REGION      AMERICA
Q3  SEGMENT     FURNITURE
   DATE        1995-03-09
Q4  DATE        1994-06-01
Q5  REGION      AMERICA
   DATE        1994-01-01
Q6  DATE        1994-01-01
   DISCOUNT    0.07
   QUANTITY    24
Q7  NATION1     PERU
   NATION2     ARGENTINA
Q8  NATION      ARGENTINA
   REGION      AMERICA
   TYPE        STANDARD POLISHED TIN
Q9  COLOR       blush
Q10 DATE        1994-03-01
```

```
Q11 NATION      CANADA
   FRACTION    0.0000000100
Q12 SHIPMODE1   SHIP
   SHIPMODE2   FOB
   DATE        1995-01-01
Q13 WORD1       express
   WORD2       deposits
Q14 DATE        1996-11-01
Q15 DATE        1995-09-01
Q16 BRAND       Brand#43
   TYPE        MEDIUM PLATED
   SIZE1       26
   SIZE2       42
   SIZE3       20
   SIZE4       10
   SIZE5       15
   SIZE6       19
   SIZE7       16
   SIZE8       36
Q17 BRAND       Brand#33
   CONTAINER   LG CAN
Q18 QUANTITY    313
Q19 BRAND1      Brand#23
   BRAND2      Brand#21
   BRAND3      Brand#54
   QUANTITY1   5
   QUANTITY2   17
   QUANTITY3   20
Q20 COLOUR      indian
   DATE        1994-01-01
   NATION      CANADA
Q21 NATION      INDONESIA
Q22 I1          32
   I2          16
   I3          10
   I4          22
   I5          12
   I6          31
   I7          15

Throughput Stream = 7   Seed = 511093325
-- TPC TPC-H Parameter Substitution (Version 2.1.1b)
-- using 511093325 as a seed to the RNG
Q1  DELTA       90
Q2  SIZE        26
   TYPE        TIN
   REGION      MIDDLE EAST
Q3  SEGMENT     MACHINERY
   DATE        1995-03-26
Q4  DATE        1997-01-01
Q5  REGION      ASIA
   DATE        1994-01-01
Q6  DATE        1994-01-01
   DISCOUNT    0.04
   QUANTITY    25
Q7  NATION1     INDONESIA
   NATION2     PERU
Q8  NATION      PERU
   REGION      AMERICA
   TYPE        STANDARD BURNISHED NICKEL
Q9  COLOR       azure
Q10 DATE        1994-12-01
Q11 NATION      MOZAMBIQUE
   FRACTION    0.0000000100
Q12 SHIPMODE1   MAIL
   SHIPMODE2   SHIP
   DATE        1997-01-01
Q13 WORD1       express
   WORD2       deposits
Q14 DATE        1997-02-01
Q15 DATE        1993-05-01
Q16 BRAND       Brand#23
   TYPE        ECONOMY BRUSHED
   SIZE1       17
   SIZE2       42
   SIZE3       4
   SIZE4       35
   SIZE5       40
   SIZE6       29
   SIZE7       3
   SIZE8       41
Q17 BRAND       Brand#35
   CONTAINER   MED CASE
Q18 QUANTITY    315
Q19 BRAND1      Brand#35
   BRAND2      Brand#14
   BRAND3      Brand#43
   QUANTITY1   10
   QUANTITY2   18
   QUANTITY3   27
Q20 COLOUR      seashell
   DATE        1993-01-01
   NATION      CHINA
Q21 NATION      ARGENTINA
Q22 I1          23
   I2          14
   I3          24
   I4          21
   I5          12
   I6          10
   I7          27

Throughput Stream = 8   Seed = 511093326
-- TPC TPC-H Parameter Substitution (Version 2.1.1b)
-- using 511093326 as a seed to the RNG
Q1  DELTA       98
Q2  SIZE        14
   TYPE        COPPER
   REGION      AMERICA
Q3  SEGMENT     BUILDING
   DATE        1995-03-11
Q4  DATE        1994-10-01
Q5  REGION      EUROPE
   DATE        1994-01-01
Q6  DATE        1994-01-01
   DISCOUNT    0.02
   QUANTITY    24
Q7  NATION1     ARGENTINA
   NATION2     INDONESIA
Q8  NATION      INDONESIA
   REGION      ASIA
   TYPE        PROMO BRUSHED NICKEL
Q9  COLOR       wheat
Q10 DATE        1993-09-01
Q11 NATION      EGYPT
   FRACTION    0.0000000100
Q12 SHIPMODE1   TRUCK
   SHIPMODE2   SHIP
   DATE        1997-01-01
Q13 WORD1       express
   WORD2       deposits
Q14 DATE        1997-06-01
Q15 DATE        1995-12-01
Q16 BRAND       Brand#53
   TYPE        SMALL ANODIZED
   SIZE1       5
   SIZE2       49
   SIZE3       39
   SIZE4       11
   SIZE5       3
```

```
   SIZE6       22
   SIZE7       26
   SIZE8       27
Q17 BRAND      Brand#31
   CONTAINER   MED JAR
Q18 QUANTITY   312
Q19 BRAND1     Brand#32
   BRAND2      Brand#42
   BRAND3      Brand#43
   QUANTITY1   6
   QUANTITY2   19
   QUANTITY3   24
Q20 COLOUR     deep
   DATE        1996-01-01
   NATION      INDIA
Q21 NATION     ROMANIA
Q22 I1         18
   I2          17
   I3          13
   I4          22
   I5          12
   I6          19
   I7          14

Throughput Stream = 9   Seed = 511093327
-- TPC TPC-H Parameter Substitution (Version 2.1.1b)
-- using 511093327 as a seed to the RNG
Q1  DELTA      106
Q2  SIZE       1
   TYPE        STEEL
   REGION      MIDDLE EAST
Q3  SEGMENT    MACHINERY
   DATE        1995-03-28
Q4  DATE       1997-05-01
Q5  REGION     AFRICA
   DATE        1995-01-01
Q6  DATE       1995-01-01
   DISCOUNT    0.07
   QUANTITY    24
Q7  NATION1    CHINA
   NATION2     ARGENTINA
Q8  NATION     ARGENTINA
   REGION      AMERICA
   TYPE        PROMO PLATED NICKEL
Q9  COLOR      steel
Q10 DATE       1994-06-01
Q11 NATION     PERU
   FRACTION    0.0000000100
Q12 SHIPMODE1  RAIL
   SHIPMODE2   SHIP
   DATE        1997-01-01
Q13 WORD1      express
   WORD2       deposits
Q14 DATE       1997-09-01
Q15 DATE       1993-09-01
Q16 BRAND      Brand#44
   TYPE        LARGE PLATED
   SIZE1       28
   SIZE2       18
   SIZE3       23
   SIZE4       47
   SIZE5       33
   SIZE6       26
   SIZE7       34
   SIZE8       32
Q17 BRAND      Brand#43
   CONTAINER   MED CAN
Q18 QUANTITY   314
Q19 BRAND1     Brand#34
   BRAND2      Brand#35
   BRAND3      Brand#42
   QUANTITY1   1
   QUANTITY2   20
   QUANTITY3   20
Q20 COLOUR     pale
   DATE        1995-01-01
   NATION      UNITED KINGDOM
Q21 NATION     IRAQ
Q22 I1         15
   I2          26
   I3          25
   I4          33
   I5          34
   I6          21
   I7          18
```

Appendix - D
Driver Source Code

D.1 tpcdbatch.h

```
/** 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"

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

#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
                                  specified 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)
#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))                /*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
*/
```

D.2 tpcdbatch.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                  */
  int                  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    (Kwai's
request - may not be reliable - valid only if != listbufqnum)  */
  int 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 */
#ifndef LISTBUFSIZE
#define LISTBUFSIZE 0x0000000040000000L   /*   size of each buffer
- 1Gb should be enough for largest query (Q11) */
#endif
  /*  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)
/*   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
*/
static inline int compare_and_swap
(
  volatile int * const   pAtomic,
  const int              compareValue,
  const int              newValue
)
{
  int oldValue ;

  __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) */
                      );

  return (oldValue == compareValue);
}
#endif

#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 */
{
 char *ap;
 char *format;
 int rc;

 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  */

#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 + 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,&tempmask); /*
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,((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+(LISTBUFSIZE<<1)); /* locate control fields at the end of
the buffer */
        if (verbose) {
          fprintf(stderr,"acquired shared-mem listbuf, shmid %d address
%p size %lld\n",listbufshmid,listbufadr
                ,((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_time ));  /*
TIME_ACC jen*/
     }
     else   /* else get the end timestamp of previous query */
     {
       strcpy(g_struct.s_info_ptr->start_stamp, temp_time_stamp);
       start_time = temp_time_struct;

     }
     /* write the start timestamp to the file...if this is not a qualification */
     /* run, then write the seed used as well */

     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,&prevmask);  /*
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(SQLPTX)
      gettimeofday_mapped(start_time);
      rc = 0;       /* gettimeofday_mapped returns void */
    #elif defined (SQLUNIX) || defined (SQLAIX)                  /*TIMER
    jen*/
      rc = gettimeofday(start_time,NULL);
    #else
    #error Unknown operating system
    #endif

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



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


    #if defined(SQLSNI)
      status = gettimeofday(&end_time);
    #elif defined(SQLPTX)
      gettimeofday_mapped(&end_time);
      status = 0;   /* gettimeofday_mapped returns void */
    #elif defined (SQLUNIX) || defined (SQLAIX)
      status = gettimeofday(&end_time,NULL);                     /*TIMER jen*/
```

```
#elif defined (SQLOS2) || defined (SQLWINT) || defined (SQLWIN) ||
defined(SQLDOS)
  ftime(&end_time);
#else                          /** If another operating system **/
#error Unknown operating system
#endif

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

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

   }

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


void dumpCa(struct sqlca *ca)
{
   int i;
   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_stamp.tv_usec);
#elif (defined (SQLOS2) || defined(SQLWINT) || defined (SQLWIN) ||
defined(SQLDOS))
        sprintf(newtime+timeLength,".%0.3d",temp_stamp.millitm);
#else
#error Unknown operating system
#endif
        /* TIME_ACC jen stop*/
      }
    }
    else
      if (form == T_STAMP_FORM_2)
        strftime(newtime,50,"%y%m%d-%H%M%S",tp);

    return (newtime);

}

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

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

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

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

```
  double QphD = 0; */

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

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


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


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

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

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

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

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

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

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

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

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

    s_info_ptr = s_info_ptr->next;

  }

  s_info_ptr = s_info_head_ptr;

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

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

          /* jmc fix geo_mean calculation...round adjusted time properly
ROUNDING*/
          adjusted_max_time = max->elapse_time/1000;
          if (s_info_ptr->elapse_time < adjusted_max_time)
          {
            s_info_ptr->adjusted_time =
              (double)(((long)((adjusted_max_time + 0.05) * 10))/10.0);
            if (s_info_ptr->adjusted_time < 0.1)
              s_info_ptr->adjusted_time = 0.1;
          }
          /*jmc fix geo_mean calculation...round adjusted time properly
ROUNDING end*/
        }
      }
```

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

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

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


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

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


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

    fprintf(outstream, "\n\nNumber of statements: %d\n\n", s_info_ptr-
>stmt_num - 1);
    /* Calculate the arithmetic and geometric means */

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

    }


    /* print out all the appropriate information including the
       different TPC-D metrics */
    /* do not bother with this if we are in an update only stream */
    fprintf (outstream, "\nGeom.  mean queries %7.3f %15.3f\n",\
             geo_mean,adjusted_g_mean);
    if (g_struct->c_l_opt->update < 2)
    {
      fprintf (outstream, "Arith. mean queries %7.3f %15.3f\n",\
             arith_mean,adjusted_a_mean);


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

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

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

#if defined (SQLUNIX) || defined (SQLAIX)
      Ts = g_struct->stream_end_time.tv_sec - g_struct-
>stream_start_time.tv_sec + 1;
      Ts1 = (double)g_struct->stream_start_time.tv_sec +
((double)g_struct->stream_start_time.tv_usec/1000000);
      Ts2 = (double)g_struct->stream_end_time.tv_sec +
((double)g_struct->stream_end_time.tv_usec/1000000);

#elif (defined (SQLOS2) || defined(SQLWINT) || defined (SQLWIN) ||
defined(SQLDOS))
      Ts = g_struct->stream_end_time.time - g_struct-
>stream_start_time.time + 1;
      Ts1 = (double)g_struct->stream_start_time.time +
((double)g_struct->stream_start_time.millitm/1000);
      Ts2 = (double)g_struct->stream_end_time.time + ((double)g_struct-
>stream_end_time.millitm/1000);

#else
#error Unknown operating system
#endif

      /* TIME_ACC jen stop*/

     /* MARK
      ##Now do in calcmetricsp.pl##
      QppD = (3600 * g_struct->scale_factor) / adjusted_g_mean;
      QthD = (num_stmt * 3600 * g_struct->scale_factor) / Ts;
      QphD = sqrt(QppD*QthD);
     */
      /* if the decimal part has some meaningful value then print the
database size
      with decimal part; otherwise just print the integer part */
```

```
        fprintf (outstream,
            "\nGeometric mean interim value  = %10.3f   %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;  /* jenCl no of concurrent updates to be */
                      /* jenCl run at once*/
  int loop_updates = 1;       /* jenCl no of updates to be run in one  */
                      /* jenCl "concurrent" invocation.  should*/
                      /* jenCl 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_time ));  /*
TIME_ACC jen*/
        /* write the start timestamp to the file...if this is not a qualification
*/
        /* run, then write the seed used as well */
        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_time ));  /*
TIME_ACC jen*/
      /* write the start timestamp to the file...if this is not a qualification
*/
      /* run, then write the seed used as well */
      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 */
  int rows_fetch = 0;
  long sqlcode = SQL_RC_E911;             /* Temporary sqlcode to test
                                    for deadlocks */
  int max_wait = 1;                       /* Maximum number of retries
                                    for deadlock scenario */
  int col_lengths[TPCDBATCH_MAX_COLS];      /* array containing
widths of
                                    columns in returned set  */
  struct stmt_info *s_info_ptr;

  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) > LISTBUFSIZE)
       {
               fprintf(stderr,"OOOPPPSSS   parent wrote %ld bytes into
sharedmem buffer of size %ld, increase by -
DLISTBUFSIZE=0x00000000x0000000L\n"
                       ,(long)(listbufnxt - listbufcur),
(long)LISTBUFSIZE); 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_struct->sem_file));

            if (hSem == NULL)
            {
                 fprintf(stderr,
                      "CreateSemaphore (ready semaphore) failed,
GetLastError: %d, quitting\n",
                        GetLastError());
                 exit(-1);
            }
            fprintf(stderr,"Semaphore created successfully!\n");

       }
#else               /* AIX, SUN, etc. */
       /* create a semaphore key...use the name of a file that */
       /* you know exists */
       fprintf(stderr,"semfile = %s\n", g_struct->update_num_file);
       semkey = ftok(g_struct->update_num_file,'J');
       if (g_struct->c_l_opt->intStreamNum == 0)
       /* RUNPOWER */
       {

            if ( (semid =
semget(semkey,2,IPC_CREAT|S_IRUSR|S_IWUSR)) < 0)
            {
                  fprintf(stderr,
                       "Throughput can't get initial semaphore! semget
failed errno = %d\n",
                        errno);
                  exit(1);
            }
            /*semctl(semid,0,IPC_RMID,0);*/ /* mujib */
       }
       else
       /* THROUGHPUT */
       {

            /* TRY TO CREATE IT USING EXCL MODE */          /*
cmgarcia */
            while ( (semid =
semget(semkey,1,IPC_CREAT|IPC_EXCL|S_IRUSR|S_IWUSR)) < 0)
            {
                  if (errno == EEXIST)
                  {
                       /* IT ALREADY EXISTS */
                       if (verbose)
                       {
                            fprintf(stderr,
```

```
                                "Throughput can't get initial semaphore!
semget failed errno = EEXIST...retrying\n");
                        }
                        errno = 0;

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

                        /* REMOVE THE SEMAPHORE */
                        if (semctl (semid, 1, IPC_RMID) < 0)
                        {
                                fprintf(stderr,
                                "Throughput can't remove initial
semaphore! semget failed errno = %d\n",
                                errno);
                                exit(1);
                        }
                }
                else
                {
                        fprintf(stderr,
                                "Throughput can't get initial semaphore!
semget failed errno = %d\n",
                                errno);
                        exit(1);
                }
        } /* IF WE COULDN'T TRY AGAIN */


        /* jlr
        if ( (semid =
semget(semkey,1,IPC_CREAT|S_IRUSR|S_IWUSR)) < 0)
        {
                fprintf(stderr,
                        "Throughput can't get initial semaphore! semget
failed errno = %d\n",
                        errno);
                exit(1);
        }
        */


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

        if (verbose)
        {
                fprintf(stderr,
                        "insert: semkey = %ld, semid = %d, file = %s,
value = %d\n",
                        semkey,semid,g_struct->update_num_file,
                        (g_struct->c_l_opt->intStreamNum * -1));
        }
    }


#endif
}

/*throughput update */
void throughput_wait(struct global_struct *g_struct)
{
#ifndef SQLWINT
  int           semid;          /* semaphore for controlling UFs*/
  key_t         semkey;         /* key to generate semid */
#else
  HANDLE        hSem;
  int           j;
  int           SemTimeout = 600000;    /* Des time out period of 1
minute */
#endif


#ifdef SQLWINT
    hSem = open_semaphore(g_struct, THROUGHPUT_SEM);
    for (j = 0; j < g_struct->c_l_opt->intStreamNum; j++)
    {
        if (verbose)
          fprintf(stderr,"About to wait again ...\n");
        if (WaitForSingleObject(hSem, INFINITE) == WAIT_FAILED)
        {
            fprintf(stderr,
              "WaitForSingleObject (hSem) failed on stream %d,
error: %d, quitting\n",
                j, GetLastError());
            exit(-1);
        }
        if (verbose)
            fprintf(stderr,"Streams to wait for  %d\n", j);
    }
    fprintf(stderr,"finished waiting on stream semaphore! Ready to run
updates!\n");
    /* close the semaphore handle */
    if (! CloseHandle(hSem)) {
      fprintf(stderr, "Close Sem failed - Last Error: %d\n",
GetLastError());
      /* no exit here */
    }
#else
    semid = open_semaphore(g_struct);
    /* call the sem_op routine to decrement the semaphore by */
    /* however many streams .... by calling this function with*/
    /* a negative number, this stream is forced to wait until */
    /* the semaphore gets back to 0 */
    if (sem_op(semid, 0, (g_struct->c_l_opt->intStreamNum * -1)) !=
0)
    {                                       /*jenSEM*/
        fprintf(stderr,
          "Failure to wait on throughput semaphone for %d
streams\n",
            g_struct->c_l_opt->intStreamNum);
        exit(1);
    }                                       /*jenSEM*/
    fprintf(stderr,"finished waiting on stream semaphore! Ready to run
updates!\n");
    semctl(semid,0,IPC_RMID,0);  /* we've finished waiting, now */
                /* remove the semaphore */
#endif
}

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


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

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

   strcpy (semfile, g_struct->update_num_file);

#endif

  if (g_struct->c_l_opt->update == 1)
    fprintf(stderr,"querystream 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_MODIFY_STATE |
                              SYNCHRONIZE,
                             TRUE,
                             semfile))
                  == (HANDLE)(NULL))
      {
        /*
        ** if cannot open the semaphore, wait for 0.1 second
        */
        fprintf(stderr,"Retry Open semaphore %s\n",semfile);

        Sleep(1000);
      }
```

```
	return hSem;
}

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

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

	semkey = ftok(g_struct->update_num_file,'J');
	while ((semid = semget(semkey,num,0)) < 0)
	{
		if (errno == ENOENT)
		{
			sleep(2);
			fprintf(stderr,"cleanUp: looping for access to semaphore
stream %d  ",
				g_struct->c_l_opt->intStreamNum);
			fprintf(stderr,"semkey=%ld semid = %d
file=%s\n",semkey,semid,
				g_struct->update_num_file);
		}
		else
		{
			fprintf(stderr,"query stream %d semget failed errno =
%d\n",
				g_struct->c_l_opt->intStreamNum,errno);
			exit(1);
		}
	}
	return semid;
}
#endif
```

D.3 tpcdUF.sqc

```
#define UF1DEBUG
#define UF2DEBUG

#if (defined(SQLPTX) && defined(SQLSUN))
#define exit(rc) _exit(rc)
#else
#define exit(rc) exit(rc)
#endif /* SQLPTX & SQLSUN*/


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

#include "sqlca.h"
#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_struct *)NULL));
/* TIME_ACC jen*/


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

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



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

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

#ifdef UF1DEBUG
  fprintf (myoutstream, "Before While_a Chunk= %d \n",UF_chunk);
  fflush(myoutstream);
#endif
    /* Loop to handle any deadlocks */
    while (sqlcode == SQL_RC_E911 && maxwait <= MAXWAIT &&
rc==0)
    {
      sqlcode = 0;
#ifdef UF1DEBUG
  fprintf (myoutstream, "in loop before orders exec sql\n");
  fflush(myoutstream);
#endif
      EXEC SQL INSERT INTO TPCD.ORDERS
            SELECT
O_ORDERKEY,O_CUSTKEY,O_ORDERSTATUS,O_TOTALPRICE,

O_ORDERDATE,O_ORDERPRIORITY,O_CLERK,O_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_QUAN
TITY,
          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\n",
                updatePair, UF_chunk,
                T_STAMP_1LEN,T_STAMP_1LEN,  /* TIME_ACC jen*/
                get_time_stamp(T_STAMP_FORM_1,
                        (Timer_struct *)NULL));  /* TIME_ACC jen*/

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

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

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

    goto UF1_exit;
  }

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


UF1_exit:
```

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

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

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

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

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

    sleep(1);
  }

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



/*****************************************************************/
/* UF2 child                                                     */
/*****************************************************************/
void runUF2_fn ( int updatePair, int thisConcurrentDelete, int
numChunks, char *dbname, char *userid, char *passwd )
{
  int rc = 0;
  long sqlcode;
  int maxwait;
  int startChunk = thisConcurrentDelete*numChunks; /* where do we
start? */
  long deletedLineitems = 0; /*kmw*/
  long deletedOrders = 0;    /*kmw*/
  long savedDeletedLineitems = 0; /*kbs*/

#ifndef SQLWINT
  int              su_semid; /* semaphore for controlling split updates*/
  key_t             su_semkey; /* key to generate semid */
#else
  HANDLE              su_hSem;
  char              UF2_semfile[256];
#endif

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

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


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

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

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

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

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

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

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

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

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

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

      EXEC SQL DELETE FROM TPCD.LINEITEM
            WHERE L_ORDERKEY IN
              (SELECT O_ORDERKEY FROM
TPCDTEMP.ORDERS_DEL
               WHERE APP_ID = :UF_chunk);
               /*AND O_ORDERKEY IN
                 (SELECT O_ORDERKEY FROM TPCD.ORDERS
                  WHERE 12*(YEAR(O_ORDERDATE)-
1992)+MONTH(O_ORDERDATE)-01 = :month));*/
      if (sqlca.sqlcode < 0)
        sqlcode = error_check();

      if (sqlcode == SQL_RC_E911)
      {       /* we've hit a deadlock */
        fprintf (myoutstream,
          "\nA deadlock detected while deleting from LINEITEM: update
pair %d set %d chunk %d.  Retrying.\n",
          updatePair, thisConcurrentDelete, UF_chunk);
        dumpCa(&sqlca);
        SleepSome(UF_DEADLOCK_SLEEP);
        maxwait++;      /* jen DEADLOCK */
      }
      else if (sqlcode < 0)
      {
        fprintf (myoutstream,"\n%s\n", stmt_str.data);
        fprintf (myoutstream,"\nsqlcode %d occurred deleting from
TPCD.LINEITEM\n", sqlca.sqlcode);
        dumpCa(&sqlca);
        fprintf (myoutstream,
              "for update pair number %d set %d chunk %d..Exiting\n",
                updatePair, thisConcurrentDelete,UF_chunk);
        rc=-1;
      }
      else
      {
        /* accumulate the number of row deleted */
        savedDeletedLineitems = sqlca.sqlerrd[2];   /*kbs*/

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

        /* delete the orders now */

        EXEC SQL DELETE FROM TPCD.ORDERS
        WHERE O_ORDERKEY IN
          (SELECT O_ORDERKEY FROM TPCDTEMP.ORDERS_DEL
WHERE APP_ID = :UF_chunk);
           /*AND 12*(YEAR(O_ORDERDATE)-
1992)+MONTH(O_ORDERDATE)-01 = :month;*/

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

        if (sqlcode == SQL_RC_E911)
        {       /* we've hit a deadlock */
#ifdef UF2DEBUG
  fprintf (myoutstream, "orders deadlocked\n");
  fflush(myoutstream);
#endif
          fprintf (myoutstream,
          "\nA deadlock detected while deleting from ORDERS: update
pair %d set %d chunk %d.  Retrying.\n",
          updatePair, thisConcurrentDelete, UF_chunk);
          dumpCa(&sqlca);
          SleepSome(UF_DEADLOCK_SLEEP);
          maxwait++;      /* jen DEADLOCK */
        }
        else if (sqlcode < 0)
        {
#ifdef UF2DEBUG
  fprintf (myoutstream, "orders failed\n");
  fflush(myoutstream);
#endif
          fprintf (myoutstream,"\nAn error %d occurred deleting from
TPCD.ORDERS\n",sqlca.sqlcode);
          dumpCa(&sqlca);
          fprintf (myoutstream,"for update pair number %d set %d
chunk %d..Exiting\n",
                updatePair, thisConcurrentDelete,UF_chunk);
          rc=-1;
        }
        else
        {
#ifdef UF2DEBUG
  fprintf (myoutstream, "orders succeeded\n");
  fflush(myoutstream);
#endif
          /* accumulate the number of row deleted */
          /* Order count ONLY updated if both orders and lineitem */
          /* go through */
          deletedLineitems += savedDeletedLineitems;   /* kbs */
          deletedOrders += sqlca.sqlerrd[2];
          rc=0;
          EXEC SQL COMMIT WORK;
          error_check();
#ifdef UF2DEBUG
```

```
            /* report the number of rows deleted */
            fprintf(myoutstream, "    interim %ld rows for chunk %d from
TPCD.ORDERS at %*.*s\n",

deletedOrders,UF_chunk,T_STAMP_1LEN,T_STAMP_1LEN,  /*
TIME_ACC jen*/
                    get_time_stamp(T_STAMP_FORM_1,(Timer_struct
*)NULL));  /* TIME_ACC jen*/
            fprintf(myoutstream, "    interim %ld rows for chunk %d from
TPCD.LINEITEM at %*.*s\n",

deletedLineitems,UF_chunk,T_STAMP_1LEN,T_STAMP_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_STATE |
                          SYNCHRONIZE,
                          TRUE,
                          UF2_semfile))
        == (HANDLE)(NULL)) {
      /*
      ** if cannot open the semaphore, wait for 0.1 second
      */
      fprintf(stderr,"Retry Open semaphore %s\n", UF2_semfile);

      SleepSome(1);
    }

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

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

/* 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");
}


   /*  tpcd_load_staging uses db2Load api to load a staging table */
int tpcd_load_staging(int staging_id     /*    what to load            *
                                specified 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;
  /****   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 */
  EXEC SQL CONNECT TO :UF_dbname;
  rc = error_check();
  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) */
#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 */
      && (kx = atoi(charptr))                   /*  non-zero  */
      && (charptr = getenv("TPCD_LN_PER_PN"))   /*  specified */
      && (tx = atoi(charptr))                   /*  non-zero  */
     )
      kx *= tx;                             /*  total number of logical nodes */
  else kx = 0;
  if (kx < 100)
    kx = 100;                               /*  minimum of 100 to be safe */
  else
  {
    kx += 10;                               /*    make it ...           */
    kx *= 2;                                /*    ... bigger            */
  }
  kx *= (sizeof(db2LoadAgentInfo));         /*  size required for kx 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 = 100;
    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 */
    rc = db2Load(
#ifdef SQLZ_CURRENT_RELEASE
        SQLZ_CURRENT_RELEASE
#else
        db2Version820
#endif
       ,&loadParms ,&sqlca);

    /* 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 */
   EXEC SQL CONNECT RESET;
   sqlcode = error_check();
  if (rc == 0)
    rc = sqlcode;
  }
  return rc;
}
```

D.4 Makefile

```
DB=tpcd

BASE=$(HOME)/sqllib
COMPILE_FLAGS= -c -DSQLAIX -I$(BASE)/include -O2 -qmaxmem=-
1 -q64 -DLISTBUFSIZE=0x0000000018000000L -
DTPCD_PROGRESS_FILE='"/tpc/tpcd/tpcd/auditruns/CURRUN/Progr
essFile"' -DTPCD_PROGRESS_LISTBUF  -DTPCD_PLOAD_API  -
DSLAVE_SEM
LINK_FLAGS= -o $@ -L$(BASE)/lib -ldb2 -q64 -lm
COMPILER=xlc
LIB_LINKER=ld
LIB_LINK_FLAGS= -o $@ -H512 -T512 -bE:$@.exp -L$(BASE)/lib -
ldb2 -lc

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

all : tpcdbatch

tpcdbatch.c : tpcdbatch.sqc
       @echo 'connect to $(DB) \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 $(DB) \n prep tpcdUF.sqc BINDFILE
PACKAGE ISOLATION RR 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
```

D.5 runpower

```
:   # -*-Perl-*-
eval 'exec perl -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");
}

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");

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


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


if (($ret2 == 0) && ($ret == 0))
{
    print "Power stream completed succesfully.\n";
}
else
{
    print "Power stream failed. ret=$ret\n";
}

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

    #clean up
}
if ($gatherstats eq "on")
{
    # kill vm io and net stats
    if ($platform eq "aix" || $platform eq "sun" || $platform eq "ptx" ||
        $platform eq "hp")
    {
      system("perl killstats &");
    }
    else
    {
        print "Stats gather not set up for current platform $platform\n";
    }
}

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

if ( $product eq "pe" )
{
    system("db2 \"connect to $dbname\"; db2 \"select
name,creator,valid,unique_id,isolation from sysibm.sysplan where
name like 'TPCD%'\";db2 connect reset;db2 terminate  >>
$runDir${delim}miso$runNum");
}
else
{
    &verifyTPCDbatch("$misofile","$dbname");
}
if ( $RealAudit ne "yes" )
{
    $curTs = `perl gettimestamp "short"`;
    # grab the db and dbm snapshot before we deactivate
    system("db2 get snapshot for all on $dbname >
$runDir${delim}dbrun$runNum.snap.$curTs");
    system("db2 get snapshot for database manager >>
$runDir${delim}dbrun$runNum.snap.$curTs");
}

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

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

# generate the mpinter?.metrics file in the run directory
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)
&cat("$runDir${delim}mpqinter*","$runDir${delim}mpinter$runNum.metr
ics");

if ($runUF eq "UF") {

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

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

1;

sub getConfig
{
  $testtype=$_[0];
  print "Getting database configuration.\n";
  $dbtunefile="$runDir${delim}m${testtype}dbtune${runNum}";
  open(DBTUNE, ">$dbtunefile") || die "Can't open $dbtunefile: $!\n";
  $timestamp=`perl gettimestamp "long"`;
  print DBTUNE "Database and Database manager configuration taken
at : $timestamp";
  # get library info for tsumami
  print DBTUNE "\nLIBRARYS\n--------\n";
  close(DBTUNE);
  system("ls -l /install/libs/*.current >> $dbtunefile");
  system("ls -lL /install/libs/*.current >> $dbtunefile");
```

```
  #system("ls -l ~/tpcd/custom/bpvars.cfg >> $dbtunefile");
  system("ls -l `db2set | grep DB2BPVARS | cut -b 11-` >>
$dbtunefile");
  #system("cat ~/tpcd/custom/bpvars.cfg >> $dbtunefile");
  system("cat `db2set | grep DB2BPVARS | cut -b 11-` >> $dbtunefile");

  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}";
      system("rah 'GetSysConfig $runDir${delim}sysconfig'");
      system("rah 'runas root ${delim}usr${delim}sbin${delim}vmo -a' >>
$ostunefile");
      system("rah 'runas root ${delim}usr${delim}sbin${delim}ioo -a' >>
$ostunefile");
      system("rah 'runas root ${delim}usr${delim}sbin${delim}no -a' >>
$ostunefile");
      system("rah 'runas root ${delim}usr${delim}bin${delim}svmon -G'
>> $ostunefile");
      system("rah '${delim}usr${delim}bin${delim}ipcs -ma' >>
$ostunefile");
      system("rah 'env' >> $ostunefile");
  }
  else
  {
      print "OS parameters retrieval not supported for $platform \n";
  }

}

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

D.6 runthroughput

```
:   # -*-Perl-*-
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];
  }
}

if (length($ENV{"TPCD_AUDIT_DIR"}) <= 0)
{
  die "TPCD_AUDIT_DIR environment variable not set\n";
}
if (length($ENV{"TPCD_RUN_DIR"}) <= 0)
{
  die "TPCD_RUN_DIR environment variable not set\n";
}
if (length($ENV{"TPCD_DBNAME"}) <= 0)
{
  die "TPCD_DBNAME environment variable not set\n";
}
if (length($ENV{"TPCD_RUNNUMBER"}) <= 0)
{
  die "TPCD_RUNNUMBER environment variable not set\n";
}
if (length($ENV{"TPCD_NUMSTREAM"}) <= 0)
{
  die "TPCD_NUMSTREAM environment variable not set\n";
}
if (length($ENV{"TPCD_RUN_ON_MULTIPLE_NODES"}) <= 0)
{
  die "TPCD_RUN_ON_MULTIPLE_NODES environment variable not
set\n";
}
if (length($ENV{"TPCD_SF"}) <= 0)
{
  die "TPCD_SF environment variable not set\n";
}
if (length($ENV{"TPCD_PRODUCT"}) <= 0)
{
  die "TPCD_PRODUCT environment variable not set\n";
}
if (length($ENV{"TPCD_PLATFORM"}) <= 0)
{
  die "TPCD_PLATFORM environment variable not set\n";
}
if (length($ENV{"TPCD_AUDIT"}) <= 0)
{
```

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

#set up local variables
$runNum=$ENV{"TPCD_RUNNUMBER"};
$numStream=$ENV{"TPCD_NUMSTREAM"};
$runDir=$ENV{"TPCD_RUN_DIR"};
$auditDir=$ENV{"TPCD_AUDIT_DIR"};
$dbname=$ENV{"TPCD_DBNAME"};
$sf=$ENV{"TPCD_SF"};
$product=$ENV{"TPCD_PRODUCT"};
$multinode=$ENV{"TPCD_RUN_ON_MULTIPLE_NODES"};
$platform=$ENV{"TPCD_PLATFORM"};
$delim=$ENV{"TPCD_PATH_DELIM"};
$RealAudit=$ENV{"TPCD_AUDIT"};
$inlistmax=$ENV{"TPCD_INLISTMAX"};
$gatherstats=$ENV{"TPCD_GATHER_STATS"};
$logDir=$ENV{"TPCD_LOG_DIR"};
$rootPriv=$ENV{"TPCD_ROOTPRIV"};
$mode=$ENV{"TPCD_MODE"};

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

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

# return 1 if the given pattern(parameter $_[0]) matches any file
sub existfile {
  if ($platform eq "aix" || $platform eq "sun" || $platform eq "ptx")
  {
    `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;
}

$misofile="$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" ||
$platform eq "hp")
{
    system("$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\"");
}

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

```
}

if ( $multinode ne "yes" )
{
  # we are running the query streams and update stream from the same
node or
  # from a serial db....use semaphores for control of the update stream

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

  $loopStream=1;

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

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

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

}
else
{
  # we are running the query streams and update stream from different
nodes, use
  # files and rksh to control the update stream
  system("runthru.pe");
}

if ($platform eq "aix")
{
    # show the disks that are used/unused
    system("getdisks \"After the completion of the Throughput Test\"");
}
if ($gatherstats eq "on")
{
    # kill vm io and net stats
    if ($platform eq "aix" || $platform eq "sun" || $platform eq "ptx" ||
        $platform eq "hp")
    {
        system("perl killstats &");
    }
    else
    {
        print "Stats gather not set up for current platform $platform\n";
    }
}

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");
}

# grab the db and dbm snapshot before we deactivate
# db snapshot shows log pages written
system("getAuditRunInfo");

# 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
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)
&cat("$runDir${delim}mts*inter*","$runDir${delim}mtinter$runNum.metri
cs");

if ($runUF ne "no") {

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

if (&existfile("$runDir${delim}mp*")) {
  # generate the mplot stuff
  system("gen_mplot");
}

# 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");

#Create Catalog info
```

```
$rc = system("perl catinfo.pl p");


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

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

1;

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

  # get library info for tsumami
  print DBTUNE "\nLIBRARYS\n--------\n";
  close(DBTUNE);
  system("ls -l /install/libs/*.current >> $dbtunefile");
  system("ls -lL /install/libs/*.current >> $dbtunefile");

  #system("ls -l ~/tpcd/custom/bpvars.cfg >> $dbtunefile");
  system("ls -l `db2set | grep DB2BPVARS | cut -b 11-` >>
$dbtunefile");
  #system("cat ~/tpcd/custom/bpvars.cfg >> $dbtunefile");
  system("cat `db2set | grep DB2BPVARS | cut -b 11-` >> $dbtunefile");

  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" )
  {
     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
>> $ostunefile");
     #system("rah
'${delim}usr${delim}samples${delim}kernel${delim}vmtune >>
$ostunefile'");
     system("rah 'runas root ${delim}usr${delim}sbin${delim}vmo -a >>
$ostunefile");
     system("rah 'runas root ${delim}usr${delim}sbin${delim}ioo -a >>
$ostunefile");
     system("rah 'runas root ${delim}usr${delim}sbin${delim}no -a >>
$ostunefile");
     system("rah 'runas root ${delim}usr${delim}bin${delim}svmon -G >>
$ostunefile");
     system("rah '${delim}usr${delim}bin${delim}ipcs -ma' >>
$ostunefile");
     system("rah 'env' >> $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");
}
```

Appendix - E ACID Transaction Source Code

E.1 acid.sqc

```
#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)*10000.0)/10.0)/100.0))
*/
#define TRUNC2(d)  ((floor(nearest((d)*1000000.0)/10000.0)/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  * 10000;
   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 * 10000).  Then divide
   ** by 10000.0 and cast it to a double (eprice) for printing
   */

   EXEC SQL
     SELECT

INTEGER(DECIMAL(SUM(DECIMAL(INTEGER(INTEGER(DECIMAL
       (INTEGER(100*DECIMAL(L_EXTENDEDPRICE,20,3)), 20,3) *
       (1-L_DISCOUNT)) * (1+L_TAX)),20,3)/100.0),20,3) * 10000)
      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 / 10000.0;  /* translate to double for
printout*/

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

   EXEC SQL COMMIT;
   if (sqlca.sqlcode != 0) {
```

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

  rc = 0;
  goto Qexit;

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

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

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


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


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

  sprintf(out_fn,
"%s%cast_acidQ.out.%d",getenv("TPCD_TMP_DIR"),del(),mypid);
  out=fopen(out_fn,"a");
  gettimeofday(&tv, &tz);
  time(&timeT);
  fprintf(out,"\n---------- START of ast_acidQ tag: %d ----------
\n\n",mypid);
  fprintf(out, "ast_acidQ tag: %d, begin transaction time: (%us %06uu)
%s",
          mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));

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

  /*
  ** use the same query acidQ except don't select for specfic okey.
  ** this ensures that the ast will be used instead of the base table
  ** Have to use ast_lEprice as double since this sum is so big
  */
  EXEC SQL
    SELECT
      SUM ( L_EXTENDEDPRICE*(1-L_DISCOUNT)*(1 + L_TAX))
      into :ast_lEprice
    FROM
       TPCD.LINEITEM;

  if (sqlca.sqlcode != 0) {
    rc = sqlca.sqlcode;
    fprintf(out,"ast_acidQ **ERROR** sqlcode = %d\n",sqlca.sqlcode);
    sqlerror("ast_acidQ: select sum(l_extendedprice)", &sqlca);
    goto Qerror;
  }
  ast_eprice = ast_lEprice;  /* use ast_eprice  for printout to be
consistent*/

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

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

  rc = 0;
  goto Qexit;

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

 Qexit:
  fprintf(out,"\n---------- END of ast_acidQ tag: %d ----------\n\n",mypid);
  fflush(out);fclose(out);
  return(rc);
}
/*-------------------------------------------------------------------*/
/*       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:
    /* gjc@011214 */
    if (acid->logging) {
      gettimeofday(&tv, &tz);
      time(&timeT);
      fprintf(out,"acidT tag: %d, immediately before closing cursors:
(%us %06uu) %s",
              mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    }
    EXEC SQL CLOSE L_CURSOR;
    /* gjc@011214. */
    if (sqlca.sqlcode != 0) {
      if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
      rc = sqlca.sqlcode;
      if (acid->logging) {
        fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } else {
        fprintf(stderr,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } /* endif */
      sqlerror("acidT: Close L_CURSOR", &sqlca);
      goto Terror;
    }

    EXEC SQL CLOSE O_CURSOR;
    /* gjc@011214. */
    if (sqlca.sqlcode != 0) {
      if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
      rc = sqlca.sqlcode;
      if (acid->logging) {
        fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } else {
        fprintf(stderr,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } /* endif */
      sqlerror("acidT: Close O_CURSOR", &sqlca);
      goto Terror;
    }

    if (acid->logging) {
      gettimeofday(&tv, &tz);
      time(&timeT);
      fprintf(out,"acidT tag: %d, immediately before COMMIT: (%us
%06uu) %s",
              mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    }
    EXEC SQL COMMIT;
    if (sqlca.sqlcode != 0) {
      if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
      rc = sqlca.sqlcode;
      if (acid->logging) {
        fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } else {
        fprintf(stderr,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } /* endif */
      sqlerror("acidT: COMMIT", &sqlca);
      goto Terror;
    }
    if (acid->logging) {
      gettimeofday(&tv, &tz);
      time(&timeT);
      fprintf(out,"acidT tag: %d, after COMMIT: (%us %06uu) %s",
              mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    }
    break;
  case 3:
    if (acid->logging) {
      gettimeofday(&tv, &tz);
      time(&timeT);
      fprintf(out,"acidT tag: %d, wait before ROLLBACK: (%us %06uu)
%s",
              mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    }
    sleep(60);
  case 2:
    if (acid->logging) {
      gettimeofday(&tv, &tz);
      time(&timeT);
      fprintf(out,"acidT tag: %d, immediately before closing cursors:
(%us %06uu) %s",
              mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    }
    EXEC SQL CLOSE L_CURSOR;
    /* gjc@011214. */
    if (sqlca.sqlcode != 0) {
      if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
      rc = sqlca.sqlcode;
      if (acid->logging) {
        fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } else {
        fprintf(stderr,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } /* endif */
      sqlerror("acidT: Close L_CURSOR", &sqlca);
      goto Terror;
    }
    EXEC SQL CLOSE O_CURSOR;
    /* gjc@011214. */
    if (sqlca.sqlcode != 0) {
      if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
```

```
      rc = sqlca.sqlcode;
      if (acid->logging) {
        fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } else {
        fprintf(stderr,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } /* endif */
      sqlerror("acidT: Close O_CURSOR", &sqlca);
      goto Terror;
    }

    if (acid->logging) {
      gettimeofday(&tv, &tz);
      time(&timeT);
      fprintf(out,"acidT tag: %d, immediately before ROLLBACK: (%us
%06uu) %s",
              mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    }
    EXEC SQL rollback work;
    if (sqlca.sqlcode != 0) {
      if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
      rc = sqlca.sqlcode;
      if (acid->logging) {
        fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } else {
        fprintf(stderr,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
      } /* endif */
      sqlerror("acidT: ROLLBACK", &sqlca);
      goto Terror;
    }
    if (acid->logging) {
      gettimeofday(&tv, &tz);
      time(&timeT);
      fprintf(out,"acidT tag: %d, after ROLLBACK: (%us %06uu) %s",
              mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    }
      break;
  }

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

  rc = 0;
  goto Texit;

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

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

/*---------------------------------------------------------------*/
/*      updateQ                                                  */
/*---------------------------------------------------------------*/
int updateQ (struct update_struct *us)
{
  FILE *out;
  time_t timeT;
  struct timeval tv;
  struct timezone tz;
  int qnum;
  int rc = 0;
  int i;
  int secs2sleep;
  char buff[256];
  struct acidtype {int logging;} a, *acid;

  EXEC SQL BEGIN DECLARE SECTION;
  double    acctbal;
  double    discount;
  double    price;
  sqlint32      availqty;
  sqlint32      size;
  EXEC SQL END DECLARE SECTION;

  qnum = us->qnum;

  acid = &a;
  acid->logging= 1;

  sprintf(buff, "%s%cupdate.out",getenv("TPCD_TMP_DIR"),del());
  out=fopen(buff,"a");

  gettimeofday(&tv, &tz);
  time(&timeT);
  fprintf(out,"\n---------- START of update ----------\n\n");
  fprintf(out, "update query number: %d, begin transaction time: (%us
%06uu) %s",
          qnum, tv.tv_sec, tv.tv_usec, ctime(&timeT));

  sqlca.sqlcode = 0;
  discount = 0.25;
  price = 5000.50;
  acctbal = 10000.00;
  availqty = 10;
  size = 5;

  for (i=1; i <= 2; i++) {
    gettimeofday(&tv, &tz);
    time(&timeT);
    fprintf(out,"update query number: %d, pass %d, immediately before
UPDATE: (%us %06uu) %s",
            qnum, i, tv.tv_sec, tv.tv_usec, ctime(&timeT));

    switch (qnum)
    {
     case 1:
     {
       EXEC SQL

          UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT
+ :discount
          WHERE L_ORDERKEY IN (326,512,928,995);
        if (sqlca.sqlcode != 0) {
          rc = sqlca.sqlcode;
          if (acid->logging)
          {
            fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                    qnum, i, sqlca.sqlcode);
          }
          else
          {
```

```
          fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 1", &sqlca);
        goto Uerror;
      }
      discount = discount * (-1);
      secs2sleep = 300;
      break;
    }
    case 2:
    {
      EXEC SQL
        UPDATE TPCD.SUPPLIER set S_ACCTBAL = S_ACCTBAL +
:acctbal
        WHERE S_NAME in
('Supplier#000000647','Supplier#000000070','Supplier#000000802');
      if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
          fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        else
        {
          fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 2", &sqlca);
        goto Uerror;
      }
      acctbal = acctbal * (-1);
      secs2sleep = 90;
      break;
    }
    case 3:
    {
      EXEC SQL
        UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT
+ :discount
        WHERE L_ORDERKEY IN (260930, 402497, 457859, 509889,
58117,
                             538311, 588421, 416167, 97830, 90276);
      if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
          fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        else
        {
          fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 3", &sqlca);
        goto Uerror;
      }
      discount = discount * (-1);
      secs2sleep = 300;
      break;
    }
    case 4:
    {
      if ( i ==1 ) {
        EXEC SQL
          UPDATE TPCD.ORDERS set O_ORDERDATE =
O_ORDERDATE - 6 MONTHS
          WHERE O_ORDERKEY =  67461;
          /* WHERE O_ORDERKEY IN
(22400,28515,34338,46596,67461,92644,98307);*/
      } else {
        EXEC SQL
          UPDATE TPCD.ORDERS set O_ORDERDATE =
O_ORDERDATE + 6 MONTHS
          WHERE O_ORDERKEY =  67461;
      }
      if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
          fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        else
        {
          fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 4", &sqlca);
        goto Uerror;
      }
      secs2sleep = 300;
      break;
    }
    case 5:
    {
      EXEC SQL
        UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT
+ :discount
        WHERE L_ORDERKEY IN
(70976,566279,152897,84226,232483);
      if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
          fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        else
        {
          fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 5", &sqlca);
        goto Uerror;
      }
      discount = discount * (-1);
      secs2sleep = 300;
      break;
    }
    case 6:
    {
      EXEC SQL
        UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT
+ :discount
        WHERE L_ORDERKEY in
(33,131,161,195,229,230,231,323,353,356);
      if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
```

```
        if (acid->logging)
        {
          fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        else
        {
          fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 6", &sqlca);
        goto Uerror;
      }
      discount = discount * (-1);
      secs2sleep = 300;
      break;
    }
    case 7:
    {
      EXEC SQL
        UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT
+ :discount
        WHERE L_ORDERKEY IN
(562917,410659,16550,398401,157634,429920,45411);
      if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
          fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        else
        {
          fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 7", &sqlca);
        goto Uerror;
      }
      discount = discount * (-1);
      secs2sleep = 300;
      break;
    }
    case 8:
    {
      EXEC SQL
        UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT
+ :discount
        WHERE L_ORDERKEY IN
(129569,343591,270242,254983,98500,28963);
      if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
          fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        else
        {
          fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }

        sqlerror("update query number 8", &sqlca);
        goto Uerror;
      }
      discount = discount * (-1);
      secs2sleep = 300;
      break;
    }
    case 9:
    {
      EXEC SQL
        UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT
+ :discount
        WHERE L_ORDERKEY IN
(113509,232997,246691,379233,448162,32134);
      if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {

          fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        else
        {
          fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 9", &sqlca);
        goto Uerror;
      }
      discount = discount * (-1);
      secs2sleep = 300;
      break;
    }
    case 10:
    {
      EXEC SQL
        UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT
+ :discount
        WHERE L_ORDERKEY IN
(516487,245411,265799,253025,6914,562020);
      if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
          fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        else
        {
          fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 10", &sqlca);
        goto Uerror;
      }
      discount = discount * (-1);
      secs2sleep = 300;
      break;
    }
    case 11:
    {
      EXEC SQL
        UPDATE TPCD.PARTSUPP set PS_AVAILQTY =
PS_AVAILQTY + :availqty
        WHERE PS_PARTKEY IN
(12098,5134,13334,17052,3452,12552,1084,5797);
```

```
        if (sqlca.sqlcode != 0) {
          rc = sqlca.sqlcode;
          if (acid->logging)
          {
           fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
          }
          else
          {
           fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
          }
          sqlerror("update query number 11", &sqlca);
          goto Uerror;
        }
        availqty = availqty * (-1);
        secs2sleep = 180;
        break;
      }
      case 12:
      {
        if ( i ==1 ) {
          EXEC SQL
            UPDATE TPCD.LINEITEM set L_RECEIPTDATE =
L_RECEIPTDATE - 3 YEARS
            WHERE L_ORDERKEY IN
(33,70,195,355,677,837,960,962,1028);
        } else {
          EXEC SQL
            UPDATE TPCD.LINEITEM set L_RECEIPTDATE =
L_RECEIPTDATE + 3 YEARS
            WHERE L_ORDERKEY IN
(33,70,195,355,677,837,960,962,1028);
        }
        if (sqlca.sqlcode != 0) {
          rc = sqlca.sqlcode;
          if (acid->logging)
          {
           fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
          }
          else
          {
           fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
          }
          sqlerror("update query number 12", &sqlca);
          goto Uerror;
        }
        secs2sleep = 300;
        break;

      }
      case 13:
      {
        EXEC SQL
          UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT
+ :discount
          WHERE L_ORDERKEY IN
(263,9476,32355,34854,53445,56901);
        if (sqlca.sqlcode != 0) {
          rc = sqlca.sqlcode;
          if (acid->logging)
          {
           fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
          }
          else
          {

           fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
          }
          sqlerror("update query number 13", &sqlca);
          goto Uerror;
        }

        discount = discount * (-1);
        secs2sleep = 90;
        break;
      }
      case 14:
      {
        EXEC SQL
          UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT
+ :discount
          WHERE L_ORDERKEY IN (32,225,326,448,449,483,512);
        if (sqlca.sqlcode != 0) {
          rc = sqlca.sqlcode;
          if (acid->logging)
          {
           fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
          }
          else
          {
           fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
          }
          sqlerror("update query number 14", &sqlca);
          goto Uerror;
        }
        discount = discount * (-1);
        secs2sleep = 180;
        break;
      }
      case 15:
      {
        EXEC SQL
          UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT
+ :discount
          WHERE L_ORDERKEY IN (1,4,7,35,135,131300);
        if (sqlca.sqlcode != 0) {
          rc = sqlca.sqlcode;
          if (acid->logging)
          {
           fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
          }
          else
          {
           fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                  qnum, i, sqlca.sqlcode);
          }
          sqlerror("update query number 15", &sqlca);
          goto Uerror;
        }
        discount = discount * (-1);
        secs2sleep = 180;
        break;
```

```
    }
    case 16:
    {
      EXEC SQL
        UPDATE TPCD.PART set P_SIZE = P_SIZE + :size
        WHERE P_PARTKEY IN (4,7,15,1313);
      if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
          fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        else
        {
          fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 16", &sqlca);
        goto Uerror;
      }
      size = size * (-1);
      secs2sleep = 180;
      break;
    }
    case 17:
    {
      EXEC SQL
        UPDATE TPCD.LINEITEM set L_EXTENDEDPRICE =
L_EXTENDEDPRICE + :price
        WHERE L_ORDERKEY IN
(4065,110372,165061,265702,87138);
      if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
          fprintf(out,"update query number: %d, pass %d, **ERROR**
sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        else
        {
          fprintf(stderr,"update query number: %d, pass %d,
**ERROR** sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 17", &sqlca);
        goto Uerror;
      }
      price = price * (-1);
      secs2sleep = 90;
      break;
    }
    default:
    {
      fprintf(out,"ERROR: Invalid query number specified %d\n",
qnum);
      rc = 1;
      goto Uexit;
    }
   }

   gettimeofday(&tv, &tz);
   time(&timeT);

   if (acid->logging)
     fprintf(out,"update query number: %d, pass %d, after UPDATE:
(%us %06uu) %s",
             qnum, i, tv.tv_sec, tv.tv_usec, ctime(&timeT));
   else
     fprintf(stderr,"update query number: %d, pass %d, after
UPDATE: (%us %06uu) %s",
             qnum, i, tv.tv_sec, tv.tv_usec, ctime(&timeT));

   if ( i == 2 ) {
     gettimeofday(&tv, &tz);
     time(&timeT);
     fprintf(out,"update query number: %d, pass %d, sleeping for %d
seconds: (%us %06uu) %s",
             qnum, i, secs2sleep, tv.tv_sec, tv.tv_usec, ctime(&timeT));
     fflush(out);
     system("touch /tmp/tpcd/update.sync.sleep");

     sleep(secs2sleep);
   }

   gettimeofday(&tv, &tz);
   time(&timeT);
   fprintf(out,"update query number: %d, pass %d, immediately before
COMMIT: (%us %06uu) %s",
           qnum, i, tv.tv_sec, tv.tv_usec, ctime(&timeT));

   EXEC SQL COMMIT;
   if (sqlca.sqlcode != 0) {
     rc = sqlca.sqlcode;
     fprintf(out,"update pass %d, **ERROR** sqlcode = %d\n", i,
sqlca.sqlcode);
     sqlerror("update: COMMIT", &sqlca);
     goto Uerror;
   }
   gettimeofday(&tv, &tz);
   time(&timeT);
   if (acid->logging)
     fprintf(out,"update query number: %d, pass %d, after COMMIT:
(%us %06uu) %s",
             qnum, i, tv.tv_sec, tv.tv_usec, ctime(&timeT));
   else
     fprintf(stderr,"update query number: %d, pass %d, after
COMMIT: (%us %06uu) %s",
             qnum, i, tv.tv_sec, tv.tv_usec, ctime(&timeT));
  }

  rc = 0;
  goto Uexit;

 Uerror:
  EXEC SQL rollback work;
  if (sqlca.sqlcode != 0) sqlerror("update: ROLLBACK FAILED",
&sqlca);
  system("touch /tmp/tpcd/update.sync.sleep");

 Uexit:
  fprintf(out,"\n---------- END of update ----------\n\n");
  fflush(out);fclose(out);
  return(rc);
}

/*--------------------------------------------------------------*/
/*       connect_to_TM                                          */
/*--------------------------------------------------------------*/
void connect_to_TM( void )
{
  char *dbname_ptr;
  if ((dbname_ptr = getenv("TPCD_QUAL_DBNAME")) != NULL) {
    fprintf(stderr,"********** %s **********\n",dbname_ptr);
    strcpy (dbname, dbname_ptr);
  }
```

```
  EXEC SQL CONNECT TO :dbname IN SHARE MODE;
  if (sqlca.sqlcode <  0) {
    fprintf(stderr, "CONNECT TO %s failed SQLCODE = %d\n",
dbname, sqlca.sqlcode);
    exit(-1);
  }
  return;
}

/*---------------------------------------------------------------*/
/*       disconnect_from_TM                                      */
/*---------------------------------------------------------------*/
void disconnect_from_TM ( void )
{
  EXEC SQL CONNECT RESET;
  if (sqlca.sqlcode <  0) {
    fprintf(stderr, "DISCONNECT failed SQLCODE = %d\n",
sqlca.sqlcode);
    exit(-1);
  }
  return;
}

/*---------------------------------------------------------------*/
/*       sqlerror                                                */
/*---------------------------------------------------------------*/
void sqlerror(char *msg, struct sqlca *psqlca)
{
  FILE *err_fp;


  char  err_fn[256];


  int j,k;

  sprintf(err_fn,
"%s%cacid.sqlerrors",getenv("TPCD_TMP_DIR"),del());
  err_fp=fopen(err_fn,"a");
  fprintf(err_fp,"acid: sqlcode: %4d %s\n", psqlca->sqlcode, msg);
  fprintf(stderr,"acid: sqlcode: %4d %s\n", psqlca->sqlcode, msg);
  fflush(stderr);
  if (psqlca->sqlerrmc[0] != ' ' || psqlca->sqlerrmc[1] != ' ') {
    fprintf(err_fp,"acid: slerrmc:  ");
    for(j = 0; j < 5; j++)
    {
      for(k = 0; k < 14; k++) fprintf(err_fp,"%x ", psqlca-
>sqlerrmc[j*10+k]);
      fprintf(err_fp,"    ");
      for(k = 0; k < 14; k++) fprintf(err_fp,"%c",  psqlca-
>sqlerrmc[j*10+k]);
      fprintf(err_fp,"\n");
      if (j < 4) fprintf(err_fp,"          ");
    }
  }

  fprintf(err_fp,"acid: sqlerrp:  ");
  for(j = 0; j < 8; j++)    fprintf(err_fp,"%c", psqlca->sqlerrp[j]);
  fprintf(err_fp,"\n");

  fprintf(err_fp,"acid: sqlerrd: ");
  for(j = 0; j < 6; j++)    fprintf(err_fp," %d", psqlca->sqlerrd[j]);
  fprintf(err_fp,"\n");

  if (psqlca->sqlwarn[0] != ' ') {
    fprintf(err_fp,"acid: sqlwarn: ");
    for(j = 0; j < 8; j++)    fprintf(err_fp,"%c ", psqlca->sqlwarn[j]);
    fprintf(err_fp,"\n");
  }
  fprintf(err_fp,"\n");
  fflush(err_fp);fclose(err_fp);
}

#ifdef SQLWINT
void sleep(int sec)
{
  Sleep(sec * 10000);
}
#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 */
```

E.2 acid.h

```
/*****************************************************************************/

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

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

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

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

#define acidtime(tvsec,tvusec) tvsec*10000+tvusec/10000
#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
```

E.3 Makefile

```
DBNAME =            $(TPCD_QUAL_DBNAME)


INCLUDE =           $(HOME)/sqllib/include

#CFLAGS             =           -I$(INCLUDE) -g -Dpascal= -
DLINT_ARGS \
#             -Dfar= -D_loadds= -DSQLA_NOLINES -qflag=i:i -
qlanglvl=ansi

#LFLAGS  =          -lm -lcurses -ls -ll -ly -liconv -lbsd
CFLAGS  =          -I$(INCLUDE) -g -Dpascal= -DLINT_ARGS  \
           -DSQLA_NOLINES -qflag=i:i -qlanglvl=ansi -q64
# .. sun       -DSQLA_NOLINES
LFLAGS  =          -lm -lbsd  -q64
# sun .... LFLAGS  =             -lm


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

CC          =          cc

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

TARGET  =          $(EXEC) tsec

.SUFFIXES: .o .c .sqc .bnd

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

all:       $(TARGET)

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

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

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

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

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

Appendix - F Third Party Pricing

F.1 CDW Pricing

CDW

800.750.4239

SHOPPING CART

Your Saved Carts | Save This Cart | Edit Saved Carts | Send To An Associate

Continue to Checkout

Quantity	Product	CDW	Usually Ships	Price	Ext. Price
4	SMC EZ Switch 16 Port 10/100/1000 Layer 2 Gigabit Unmanaged Switch	587912	1-2 Weeks	$312.00	$1,248.00
Click to remove an item from your cart				Sub-Total	$1,248.00

Update | Clear Cart | Continue to Checkout

Continue Shopping | Go to CDW.com Homepage