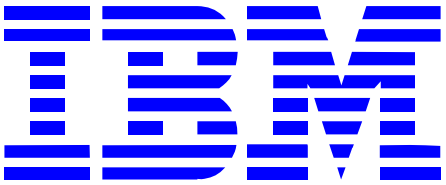


TPC Benchmark™ D Full Disclosure Report

IBM RISC System/6000 SP Model 550

using

IBM DB2 Universal Database 5.2.0

Original Submitted For Review
May 11, 1998

Modified

October 9, 1998

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

TPC Benchmark	Trademark of the Transaction Processing Performance Council
TPC-D	Trademark of the Transaction Processing Performance Council
QppD	Trademark of the Transaction Processing Performance Council
QthD	Trademark of the Transaction Processing Performance Council
QphD	Trademark of the Transaction Processing Performance Council
IBM	Trademark of International Business Machines Corporation
DB2	Trademark of IBM

First Edition May 11, 1998
Second Edition October 9, 1998

The information contained in this document has not been submitted to any formal test and is distributed on an AS IS basis without any warranty either expressed or implied. The use of this information or the implementation of any of these techniques is a customer's responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item has been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environment do so at their own risk.

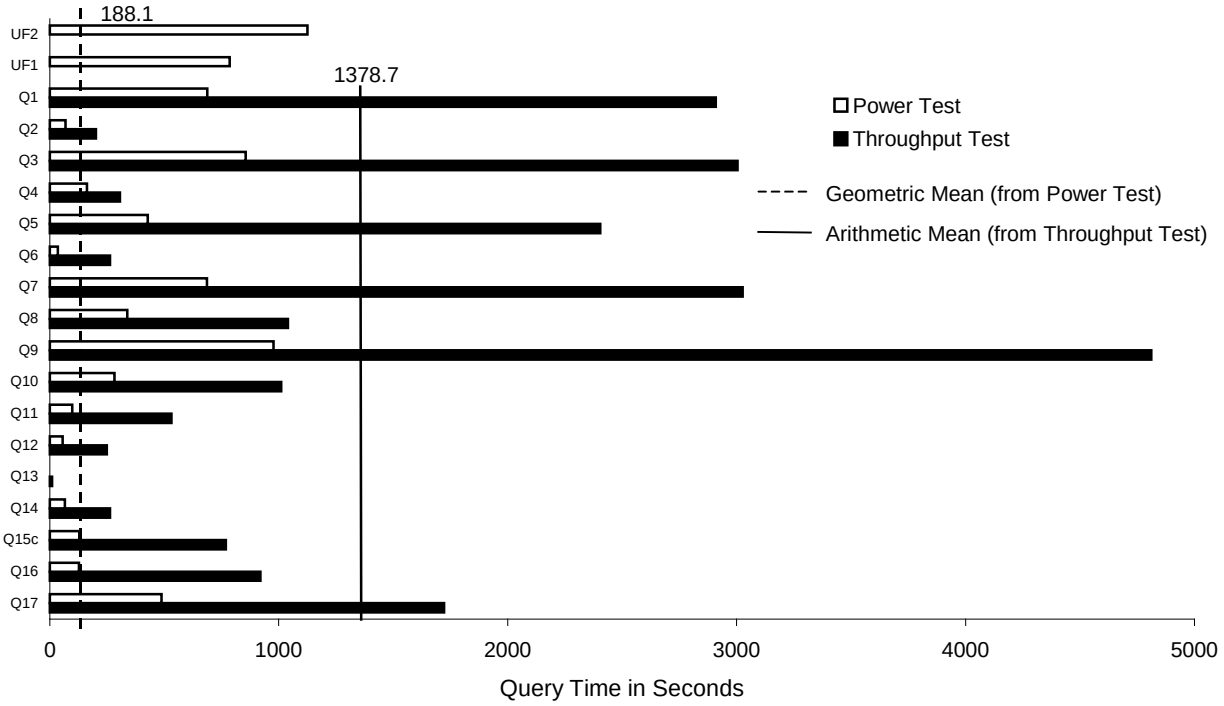
In this document, any references made to an IBM licensed program are not intended to state or imply that only IBM's licensed program may be used; any functionally equivalent program may be used.
It is possible that this material may contain references to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such products, programming, or services in your country.

All performance data contained in this publication was obtained in a controlled environment, and therefore the results which may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable date in their specific environment.

IBM	RS/6000 SP Model 550 with DB2 UDB 5.2.0	TPC-D Rev. 1.3.1 Report Date: May 11, 1998 Modified: October 9, 1998

Total System Cost	TPC-D Power	TPC-D Throughput	Price/Performance
$ 11,380,178	**19137.5** QppD@1000GB	**10661.5** QthD@1000GB	**$797** per QphD@1000GB

Database Size	Database Manager	Operating System	Other Software	Availability Date
1000 GB	**IBM DB2 Universal Database 5.2.0**	**AIX 4.2.1**	**None**	**October 31, 1998** See note in page 8



Database Load Time: 20:20:32	Total Data Storage/Database Size: 7.01	RAID: Y

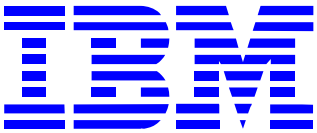
System Components

48 RS/6000 SP nodes, each node with:

4 x 332MHz PowerPC 604e CPUs (256K L2)
3GB memory
2 x 4.5GB SCSI Disk
2 x SSA disk adapters/controllers
1 x SP Switch MX Adapter

External Disk Storage:

Total Storage: 1536 x 4.5GB SSA disks

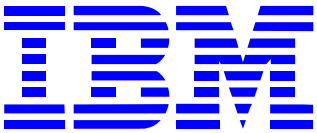


RS/6000 SP Model 550 With DB2 UDB 5.2.0

TPC-D Rev. 1.3.1

Report Date: May 11, 1998
Modified: October 9, 1998

Description	Part No.	Source	Ref. Price	Qty	Discount	Ext. Price	5-Yr. Maint.
Server Hardware							
RS/6000 SP - Tall Frame	9076-550	IBM	50,000	1		50,000	12,000
Expansion Frames	1550	IBM	50,000	2		100,000	24,000
332 MHz Thin Node	2050	IBM	24,000	48		1,152,000	506,880
4.5 GB Disk Drive – F/W 1 In High	3000	IBM	1,300	96		124,800	0
SP Switch	4011	IBM	70,000	3		210,000	60,480
SP Switch MX Adapter	4022	IBM	12,500	48		600,000	0
Memory Cards	4093	IBM	1,038	96		99,648	0
256MB DIMM Memory Modules	4110	IBM	3,840	576		2,211,840	0
PowerPC 604e, 332 MHz, 2-Way Processor Card	4320	IBM	12,000	96		1,152,000	552,960
PCI SSA Adapter	6215	IBM	3,000	96		288,000	0
	Subtotal					**5,988,288**	**1,156,320**
Control Workstation							
RS/6000 Model F30	7025-F30	IBM	12,995	1		12,995	6,720
GXT110P Graphics Adapter (PCI)	2839	IBM	340	1		340	0
8port Async Adapter	2931	IBM	1,195	1		1,195	0
Auto LANstreamer Token Ring Adapter	2979	IBM	795	1		795	0
Ethernet BNC/RJ-45 Adapter, PCI	2985	IBM	195	1		195	0
4.5 GB SCSI-2 Hard Disk Drive	3092	IBM	830	2		1,660	0
P50 Color Monitor	3612	IBM	800	1		800	0
32 MB DIMM Memory	4132	IBM	640	1		640	0
Display Cable	4217	IBM	105	1		105	0
Mouse	6041	IBM	75	1		75	0
Keyboard – English (US)	6900	IBM	140	1		140	0
	Subtotal					**18,940**	**6,720**



RS/6000 SP Model 550 with DB2 UDB 5.2.0

TPC-D Rev. 1.3.1

Report Date: May 11, 1998
Modified: October 9, 1998

Description	Part No.	Source	Ref. Price	Qty	Discount	Ext. Price	5-Yr. Maint.
Storage							
SSA Disk Subsystem with (4) 4.5 GB Drives	7133-020	IBM	19,350	96		1,857,600	921,600
4.5GB SSA Disk Drive Modules	3401	IBM	2,100	1152		2,419,200	0
Power/Cooling	3003	IBM	1,600	96		153,600	0
10.0m SSA Copper Cable	5100	IBM	75	384		28,800	0
Expansion Rack	7015-R00	IBM	3,110	16		49,760	23,808
Subtotal						**4,508,960**	**945,408**
Hardware Subtotal						**10,516,188**	**2,108,448**
Server Software							
AIX 4.2 unlimited users	5765-C34	IBM	3,660	1		3,660	0
C for AIX	5756-423	IBM	615	1		615	0
AIX 4.2.1 SP License	5756-030	IBM	0	48		0	0
AIX Parallel System Support	5765-296	IBM	0	48		0	0
DB2 UDB V5 4-way Base Program Package	5801-AAR	IBM	69,998	1		69,998	0
DB2 UDB V5 4-way Add'l Servers	5802-AAR	IBM	69,898	47		3,285,206	0
Software Subtotal						**3,359,479**	**0**
Total						**13,875,667**	**2,108,448**
Discounts							
Hardware Discount					27%	(2,839,371)	
Software Discount					33%	(1,108,628)	
MidRange Service Agreement Discount					17%		(358,436)
Extended Maint. Option Discount					17%		(297,502)
Total						**9,927,668**	**1,452,510**
5 Year Cost of Ownership						**$11,380,178**	
QphD @ 1000 GB						14284.1	
$/QphD @ 1000 GB						**$797**	

Audited By: Francois Raab, Information Paradigm Inc.

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

Note: Hardware and system software available now; DB2 UDB 5.2.0 software available by October 31, 1998.

Numerical Quantities Summary

Measurement Results:

Database Scaling (SF/Size)	=	1000
Total Data Storage/Database Size	=	7.01
Database Load Time	=	20:20:32
Query Streams for Throughput Test	=	7
TPC-D Power Metric (QppD@1000GB)	=	19137.5
TPC-D Throughput Metric (QthD@1000GB)	=	10661.5
Composite Query-per-hour Rating (QphD@1000GB)	=	14284.1
Total System Price over 5 years	=	$11,380,178
TPC-D Price/Performance Metric ($/QphD@1000GB)	=	$797

Measurement Intervals:

Measurement Interval in Throughput Test (Ts) = 40,182 seconds

Duration of Stream Execution:

Stream ID	Seed	Start Date	Start Time	End Date	End Time	Total Time
Stream0	225520782	05/04/98	23:36:08	05/05/98	01:39:26	02:03:18
Stream1	1367916679	05/05/98	02:20:12	05/05/98	08:58:31	06:38:19
Stream2	1916419193	05/05/98	02:20:12	05/05/98	08:31:47	06:11:35
Stream3	1175406391	05/05/98	02:20:12	05/05/98	08:57:17	06:37:05
Stream4	1869157532	05/05/98	02:20:13	05/05/98	08:49:24	06:29:11
Stream5	536650117	05/05/98	02:20:13	05/05/98	08:56:44	06:36:31
Stream6	722405354	05/05/98	02:20:13	05/05/98	08:50:33	06:30:20
Stream7	598060941	05/05/98	02:20:14	05/05/98	08:51:44	06:31:30
Updates	n/a	05/05/98	02:20:14	05/05/98	13:29:54	11:09:40

Timing Intervals (in seconds):

Stream ID	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
Stream0	687.4	68.3	855.4	162.8	427.4	35.4	686.3	338.7	977.7	281.7
Stream1	3222.0	235.9	2997.3	194.1	2610.7	162.8	1968.1	982.5	5665.5	1390.6
Stream2	2605.3	219.4	2875.8	477.4	1891.5	289.9	3081.5	999.0	4415.9	1056.9
Stream3	2508.3	197.8	2713.1	335.1	1789.0	316.0	3820.7	1219.5	5163.0	1086.2
Stream4	3055.1	160.2	3051.5	196.3	2119.9	165.5	3309.3	1087.9	4356.2	1278.3
Stream5	3248.2	184.7	3310.0	337.5	2735.1	292.2	2658.8	1135.6	5339.1	685.8
Stream6	2788.6	145.1	2819.2	238.2	2865.1	174.7	3577.1	1026.4	4487.6	733.2
Stream7	2934.1	265.5	3257.6	361.4	2815.9	433.3	2772.1	827.9	4250.2	846.7
Minimum	2508.3	145.1	2713.1	194.1	1789.0	162.8	1968.1	827.9	4250.2	685.8
Average	2908.8	201.2	3003.5	305.7	2403.9	262.1	3026.8	1039.8	4811.1	1011.1
Maximum	3248.2	265.5	3310.0	477.4	2865.1	433.3	3820.7	1219.5	5665.5	1390.6

Stream ID	Q11	Q12	Q13	Q14	Q15c	Q16	Q17	UF1	UF2
Stream0	98.2	56.5	1.0	66.2	128.1	127.4	487.2	786.1	1125.6
Stream1	553.9	265.7	22.8	322.4	865.8	849.3	1588.4	24708.3	1140.9
Stream2	552.4	335.7	7.4	356.0	932.4	902.9	1294.4	1255.4	1125.4
Stream3	652.6	164.4	6.1	108.2	904.5	1138.7	1700.2	1253.7	1128.3
Stream4	547.3	239.2	9.4	234.3	852.4	1074.6	1611.3	1259.5	1134.5
Stream5	436.9	280.5	3.1	187.6	277.1	813.9	1862.7	1259.4	1120.4
Stream6	435.8	221.0	10.4	285.9	821.1	734.9	2052.9	1270.6	1120.7
Stream7	542.2	236.8	3.0	343.4	727.0	923.9	1948.0	1278.0	1125.3
Minimum	435.8	164.4	3.0	108.2	277.1	734.9	1294.4	1253.7	1120.4
Average	531.6	249.0	8.9	262.5	768.6	919.7	1722.6	4612.1	1127.9
Maximum	652.6	335.7	22.8	356.0	932.4	1138.7	2052.9	24708.3	1140.9

Information Paradigm

TPC TRANSACTION PROCESSING PERFORMANCE COUNCIL

Certified Auditor

Test Sponsors: Robert Nichels
Manager, RS/6000 SP Performance
IBM Corporation, Dpt 67LB
South Road
Poughkeepsie, NY 12601

May 8, 1998

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

Platform: IBM RS/6000 SP Model 550, 48 node system
DataBase Manager: IBM DB2 Universal Database 5.2.0
Operating System: IBM AIX Version 4.2.1

The results were:

CPU's Speed	Memory	Disks	QppD@1000GB	QthD@1000GB
IBM RS/6000 SP Model 550, 48 node system				
Each node: 4 x PowerPC 604e (332 MHz)	Each node: 256 KB L2 3 GB	Each node: 32 x 4.5 GB SSA ext. 2 x 4.5 GB SCSI int.	19,137.5	10,661.5

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

- The TIME table was not used
- The input variables were generated by QGEN
- The database was populated using DBGEN
- The database was maintained by the "evolve" method
- The throughput metric was computed using the results from a 7-stream test
- The ratio between the longest and the shortest query was such that one query was adjusted
- A compliant implementation specific layer was used
- The query text was produced using compliant variants and minor modifications

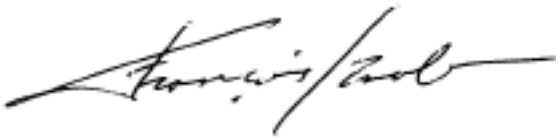
1373 North Franklin Street • Colorado Springs, CO 80903-2527 • Office : 719/473-7555 • Fax : 719/473-7554

- The database records were defined with the proper layout and size
- The database was properly scaled to 1,000GB and populated accordingly
- The database load time was correctly measured and reported
- The ACID Properties were verified and met
- The reported execution times were correctly measured and reported
- Measurement repeatability was verified
- At least 8 hours 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

1.0 Abstract

This report documents the full disclosure information required by the TPC Benchmark™ D Standard Specification Revision 1.3.1, dated Feb 12, 1998, for measurements on the IBM RS/6000 SP model 550. The software used includes the AIX 4.2.1 operating system and IBM DB2 Universal Database Enterprise Extended Edition Version 5 for AIX.

2.0 Preface

TPC Benchmark™ D Standard Specification was developed by the Transaction Processing Performance Council (TPC). It was released on May 5, 1995, and most recently revised (Revision 1.3.1) on Feb 12, 1998. This is the full disclosure report for benchmark testing of the IBM RS/6000 SP according to the TPC Benchmark™ D Standard Specification.

TPC Benchmark™ D 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-D evaluates the performance of various Decision Support Systems by the execution of sets of queries against a standard database under controlled conditions. The TPC-D queries:

- Give answers to real-world business questions;
- 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-D operations are modeled after the following business environment:

- The database is continuously available 24 hours a day, 7 days a week, for queries or updates against all tables for multiple users, except possibly during infrequent (e.g., once a month) maintenance sessions;
- The TPC-D database tracks, possibly with some delay, the state of the OLTP database through on-going updates 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-D database, the queries and the updates may be executed against the database at any time, especially in relation to each other. In addition, this mix of queries and updates is subject to specific ACIDity requirements, since queries and updates 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 updates.

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 metrics reported by TPC-D measure 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 TPC-D query processing power at the selected size (QppD@Size), and the TPC-D throughput at the selected size (QthD@Size) when queries are submitted by one or more concurrent users. The TPC-D Price/Performance metric is expressed as $/QphD@Size and is based on a composite query-per-hour rating derived from QppD and QthD. To be compliant with the TPC-D standard, all references to TPC-D results for a given configuration must include all required reporting components (see Clause 5.4.7).

The TPC-D database was 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.

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-D should not be used as a substitute for specific customer application benchmarking when critical capacity planning and/or product evaluation decisions are contemplated.

3.0 General Items

3.1 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**.

3.2 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 and system parameters and compiler options. 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. The procedure for preprocessing, binding, compiling and linking the tpcdbatch program is documented in Appendix D, “Makefile.pe” .

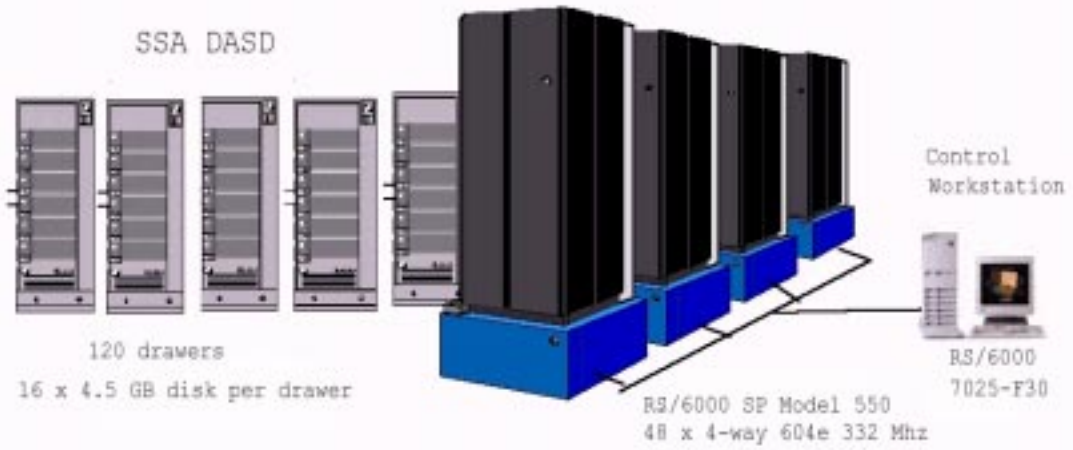
3.3 Configuration Diagrams

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

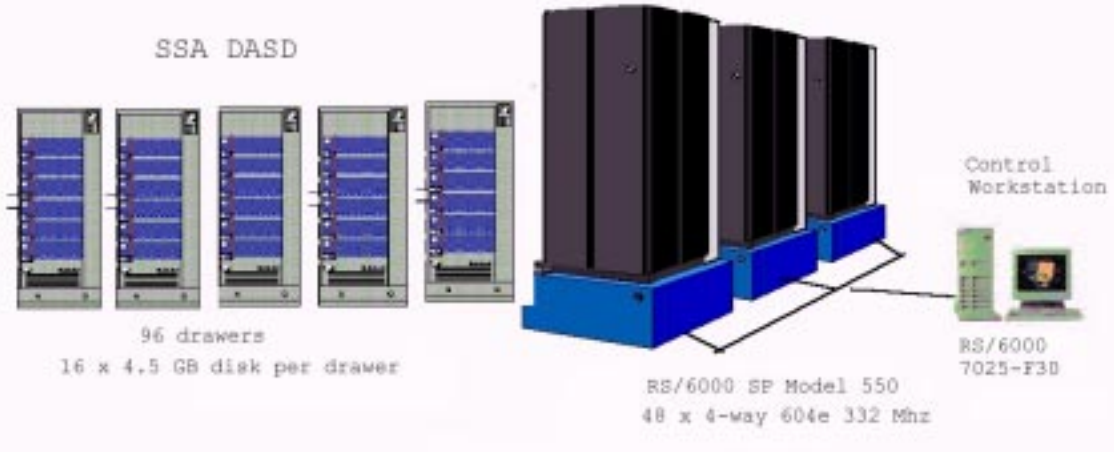
- *Number and type of processors*
- *Size of allocated memory, and any specific mapping/partitioning of memory unique to the test and type of disk units (and controllers, if applicable)*
- *Number and type of disk units (and controllers, if applicable).*
- *Number of channels or bus connections to disk 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.)*

The measured configuration consisted of four frames, two with sixteen nodes each and two with eight nodes each. The priced configuration consists of three frames of sixteen nodes each. The measured configuration has 120 * 16 = 1920 4.5GB disks while the priced configuration has 96 * 16 = 1536 4.5GB disks.

RS/6000 SP Model 550 Measured Configuration



RS/6000 SP Model 550 Priced Configuration



4.0 Clause 1: Logical Database Design

Appendix B. “Database Build Scripts” contains the programs and input files used to load the test and qualification databases. The test and qualification databases use the same table definitions, indices and partitioning methods. However, the qualification database was partitioned across four physical nodes, rather than 48 physical nodes (see the db2nodes.cfg and db2nodes.qual.cfg files in Appendix B for details on the nodes used for the qualification and test databases). Thus the scripts documented in Appendix B were used for both the qualification and test databases except that different input files were used.

The loading of the database is controlled by the buildtpcd script in Appendix B. buildtpcd, and the scripts it calls, create the database, create tablespaces, create nodegroups, create tables, set up the load configuration, load pre-generated and pre-split data, create indices, gather statistics, create constraints, and set up the runtime configuration.

4.1 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 program to load the database.

4.2 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 index definitions.

4.3 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 in all tables except for the nation and region tables.

4.4 Replication

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

No replication beyond the creation of indices (see Appendix B. “Database Build Scripts”) was used.

5.0 Clause 2: Queries and Update Functions

5.1 Query Language

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

SQL was the query language used.

5.2 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 DBGEN version 1.3.1 and QGEN version 1.3.1 were used to generate all database populations.

5.3 Substitution Parameters

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 1.3.1 was used to generate the substitution parameters.

5.4 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, "Qualification Queries & Output" contains the executable query texts and the output for each of the queries. The functional query definitions and variants used in this disclosure use the following minor query modifications.

1. Table names and view names are fully qualified. For example, the nation table is referred to as "TPCD.NATION". The "order" table is named "orders" since order is a reserved word in DB2.
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 is used at SF 1000.

5.5 Disclosure

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

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

5.6 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 was **repeatable read**

5.7 Update Functions

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

The update function is part of the implementation specific layer/driver code included in Appendix D. “Driver Source Code” .

5.8 Database Maintenance Option

The details of the database maintenance option selected (i.e., reset or evolve) must be disclosed (including source code of any non-commercial program used).

This implementation uses the evolve option.

6.0 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. Appendix E, "Acid Transaction Source Code" contains the source code for the ACID transaction and query.

6.1 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.

6.1.1 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 was executed for the Orderkey used in Step 1.
3. The total price and the extended price were retrieved for the same orderkey used in step 1 and step 2. It was verified that: EXTENDEDPRICE = EXTENDEDPRICE + ((DELTA) * (EXTENDEDPRICE/QUANTITY)), TOTALPRICE = TOTALPRICE + (COST*(1-DISCOUNT)*(1+TAX)), and that the number of records in the history table had increased by 1.

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

6.2 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.

6.2.1 Consistency Condition

A consistent state for the TPC-D 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 query was executed before and after a measurement to show that the database was always in a consistent state both initially and after a measurement.

```
SELECT DECIMAL(SUM(DECIMAL(INTEGER(INTEGER(DECIMAL
(INTEGER(100*DECIMAL(L_EXTENDEDPRICE,20,2)),20,2)*
(1-L_DISCOUNT)) * (1+L_TAX)),20,2)/100.0,20,2)
FROM TPCD.LINEITEM   WHERE L_ORDERKEY = O_ORDERKEY
```

6.2.2 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 query defined in 3.3.2 , "Consistency Condition" was run after initial database build and prior to executing the ACID transaction. The query showed that the database was in a consistent state.
After executing 100 ACID transactions the query defined in 3.3.2 , "Consistency Condition" was run again. The query showed that the database was still in a consistent state.

6.3 Isolation Requirements

6.3.1 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.

6.3.2 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.

6.3.3 Isolation Test 3

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

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

1. 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)

6.3.4 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

6.3.5 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.

6.3.6 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 update transactions affecting those tables from making progress.

1. 1st session: A transaction T1, which executes TPC-D query 1 (from TPC-D spec clause 2.3) 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 TPC-D 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.

6.4 Durability

The tested system 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.

6.4.1 System Crash

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

The system crash and memory failure tests were combined. Power to the server was turned off during the durability test. When power was restored, the system rebooted and the database was restarted. The durability success file and the HISTORY table were compared and the counts matched.

6.4.2 Memory Failure

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

See the previous section.

6.4.3 Switch Failure

Guarantee the database and committed updates are preserved across a switch failure.

Power to a switch was turned off during the durability test. The tests terminated as node communication failed.
When the switch was restored to operation, the database was restarted. The durability success file and the HISTORY table were compared and the counts matched.

6.4.4 Failure of a Durable Medium

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

The disks containing the TPC-D tables, indices and log files were mirrored. During the system crash durability test, a log disk was first disabled. The test continued uninterrupted using the remaining side of the mirror until power to test server was turned off. During the switch failure test, a data disk was first disabled. The test continued uninterrupted using the remaining side of the mirror until power to the switch was turned off.

7.0 Clause 4: Scaling and Database Population

7.1 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™ D defined tables and the number of rows for each table as they existed upon build completion:

Table	Rows
Lineitem	5,999,989,709
Orders	1,500,000,000
Customer	150,000,000
Supplier	10,000,000
Part	200,000,000
Partsupp	800,000,000
Nation	25
Region	5

7.2 Distribution of Tables and Logs

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

The RS/6000 SP System Under Test has 48 nodes. Each node has the following storage and controllers:

- 1 Small Computer Interface-2 (SCSI-2) controllers
- 2 internal 4.5GB SCSI-2 disks
- 2 Serial Storage Architecture (SSA) controllers
- 32 external 4.5GB SSA disks

All tables, indices, logs and temporary tables are mirrored.

The following table depicts the database configuration for each of the 48 nodes of the RS/6000 SP System Under Test:

Adapter	Physical Volume	Container Name	Size (MB)	Description of contents
ssa0	hdisk2,hdisk14	/temp2/temp	1,040	TPCDTEMP tablespace
ssa1	hdisk22,hdisk34	(mirror)	1,040	
ssa0	hdisk3,hdisk15	/temp3/temp	1,040	
ssa1	hdisk23 ,hdisk35	(mirror)	1,040	
ssa0	hdisk4,hdisk16	/temp4/temp	1,040	
ssa1	hdisk24,hdisk36	(mirror)	1,040	
ssa0	hdisk5,hdisk17	/temp5/temp	1,040	
ssa1	hdisk25 ,hdisk37	(mirror)	1,040	
ssa0	hdisk6,hdisk18	/temp6/temp	1,040	
ssa1	hdisk26,hdisk38	(mirror)	1,040	
ssa0	hdisk7,hdisk19	/temp7/temp	1,040	
ssa1	hdisk27,hdisk39	(mirror)	1,040	
ssa0	hdisk8,hdisk20	/temp8/temp	1,040	
ssa1	hdisk28,hdisk40	(mirror)	1,040	
ssa0	hdisk9,hdisk21	/temp9/temp	1,040	
ssa1	hdisk29,hdisk41	(mirror)	1,040	
ssa0	hdisk2	/dev/rdata2	1,840	TPCDALL tablespace
ssa1	hdisk22	(mirror)	1,840	
ssa0	hdisk3	/dev/rdata3	1,840	
ssa1	hdisk23	(mirror)	1,840	
ssa0	hdisk4	/dev/rdata4	1,840	
ssa1	hdisk24	(mirror)	1,840	
ssa0	hdisk5	/dev/rdata5	1,840	
ssa1	hdisk25	(mirror)	1,840	
ssa0	hdisk6	/dev/rdata6	1,840	
ssa1	hdisk26	(mirror)	1,840	
ssa0	hdisk7	/dev/rdata7	1,840	
ssa1	hdisk27	(mirror)	1,840	
ssa0	hdisk8	/dev/rdata8	1,840	
ssa1	hdisk28	(mirror)	1,840	
ssa0	hdisk9	/dev/rdata9	1,840	
ssa1	hdisk29	(mirror)	1,840	
ssa0	hdisk14	/dev/rdata14	1,840	
ssa1	hdisk34	(mirror)	1,840	
ssa0	hdisk15	/dev/rdata15	1,840	
ssa1	hdisk35	(mirror)	1,840	
ssa0	hdisk16	/dev/rdata16	1,840	
ssa1	hdisk36	(mirror)	1,840	
ssa0	hdisk17	/dev/rdata17	1,840	
ssa1	hdisk37	(mirror)	1,840	
ssa0	hdisk18	/dev/rdata18	1,840	
ssa1	hdisk38	(mirror)	1,840	
ssa0	hdisk19	/dev/rdata19	1,840	
ssa1	hdisk39	(mirror)	1,840	
ssa0	hdisk20	/dev/rdata20	1,840	
ssa1	hdisk40	(mirror)	1,840	
ssa0	hdisk21	/dev/rdata21	1,840	
ssa1	hdisk41	(mirror)	1,840	
ssa1	hdisk22	/dev/rdata22	1,840	
ssa0	hdisk2	(mirror)	1,840	
ssa1	hdisk23	/dev/rdata23	1,840	
ssa0	hdisk3	(mirror)	1,840	
ssa1	hdisk24	/dev/rdata24	1,840	

ssa0	hdisk4	(mirror)	1,840	
ssa1	hdisk25	/dev/rdata25	1,840	
ssa0	hdisk5	(mirror)	1,840	
ssa1	hdisk26	/dev/rdata26	1,840	
ssa0	hdisk6	(mirror)	1,840	
ssa1	hdisk27	/dev/rdata27	1,840	
ssa0	hdisk7	(mirror)	1,840	
ssa1	hdisk28	/dev/rdata28	1,840	
ssa0	hdisk8	(mirror)	1,840	
ssa1	hdisk29	/dev/rdata29	1,840	
ssa0	hdisk9	(mirror)	1,840	
ssa1	hdisk34	/dev/rdata34	1,840	
ssa0	hdisk14	(mirror)	1,840	
ssa1	hdisk35	/dev/rdata35	1,840	
ssa0	hdisk15	(mirror)	1,840	
ssa1	hdisk36	/dev/rdata36	1,840	
ssa0	hdisk16	(mirror)	1,840	
ssa1	hdisk37	/dev/rdata37	1,840	
ssa0	hdisk17	(mirror)	1,840	
ssa1	hdisk38	/dev/rdata38	1,840	
ssa0	hdisk18	(mirror)	1,840	
ssa1	hdisk39	/dev/rdata39	1,840	
ssa0	hdisk19	(mirror)	1,840	
ssa1	hdisk40	/dev/rdata40	1,840	
ssa0	hdisk20	(mirror)	1,840	
ssa1	hdisk41	/dev/rdata41	1,840	
ssa0	hdisk21	(mirror)	1,840	
ssa0/1,scsi0	all disks	/dev/rdata42	800	
ssa0/1,scsi0	all disks	(mirror)	800	
ssa0/1,scsi0	all disks	/uftbsp/uf_chunk1	600	UF_TEMP1 tablespace
ssa0/1,scsi0	all disks	(mirror)	600	
scsi0	hdisk0	/database	984	SYCATSPACE, USERSPACE1 tablespaces and logs
scsi0	hdisk1	(mirror)	984	

7.3 Mapping of Partitions/Replications

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

See Table in section 7.2 above

7.4 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 1 (mirroring) was used.

7.5 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 DBGEN version 1.3.1 was used for the database population.

7.6 Table Contents

The contents of the first 10 rows of each table in the test database must be disclosed.

Appendix C, "Test Database Table Contents" lists the contents of the first 10 rows of each table in the test database.

7.7 Database Loading

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

The Numerical Quantities Summary contains the database load times for the system tested in this full disclosure report.

7.8 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:

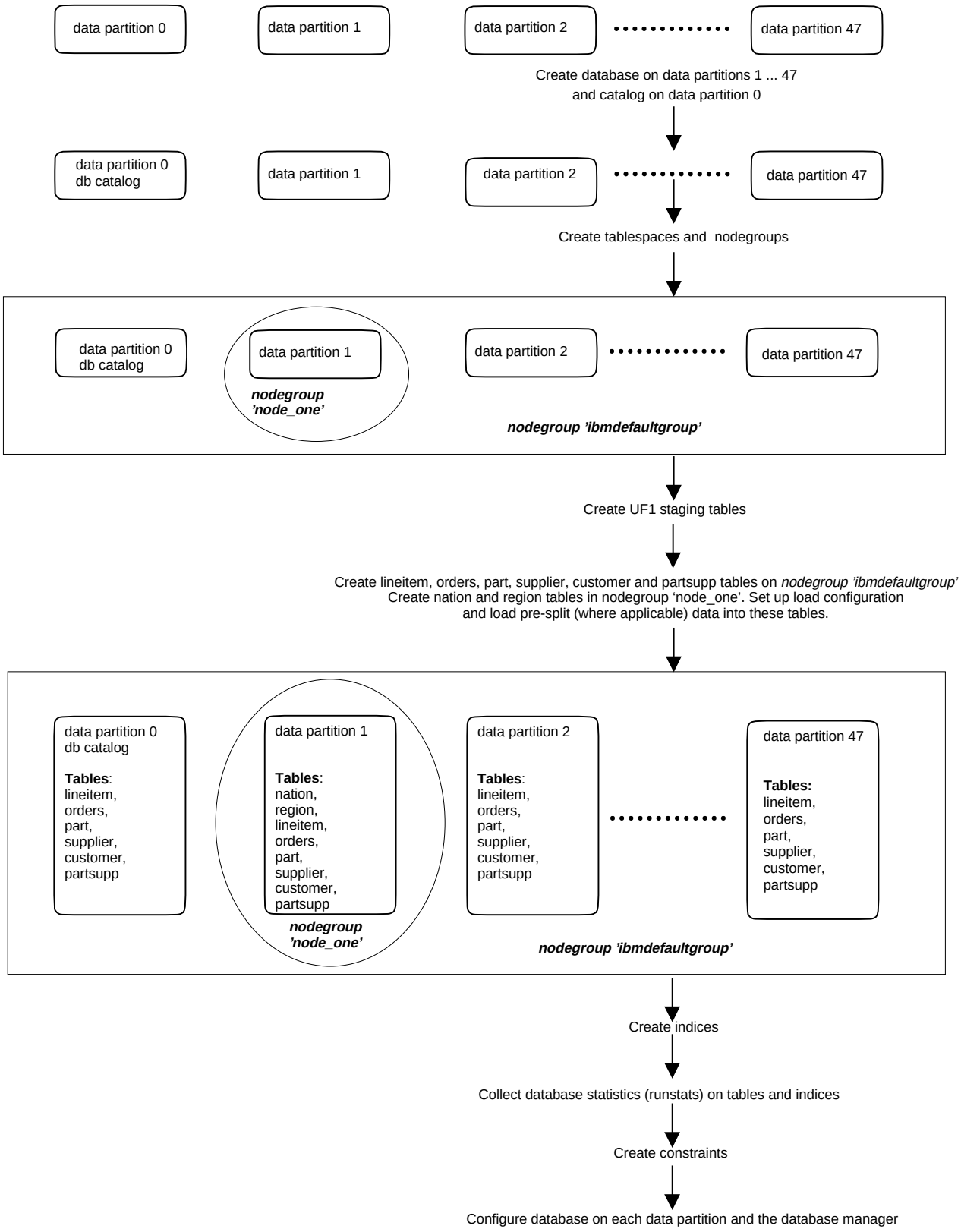
Disk Type	# of Disks	Space per Disk	Sub-Total Disk Space	Database Size	Data Storage Ratio
F/W SCSI	2 x 48	4.296 GB	412.416 GB		
SSA	32 x 48	4.296 GB	6598.656 GB		
			7011.072 GB	1000	7.01

7.9 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.

The following figure depicts the database loading procedure.

Database Load Procedure



8.0 Clause 5: Performance Metrics and Execution Rules

8.1 Power Test

8.1.1 Implementation

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

The power test is executed using the runpower script (see Appendix D). Prior to running the power test, the system is rebooted.

8.1.2 Timing Intervals

The timing intervals (see Clause 5.3.6) for each query of the measured set (i.e., the query set that follows the warmup set, see Step 4 of Clause 5.3.2.2) and for both update functions must be reported for the power test.

The Numerical Quantities Summary contains the timing intervals for the power test.

8.2 Throughput Test

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

The Numerical Quantities Summary contains the number of streams.

8.2.1 Stream Times

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

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

8.2.2 Measurement Interval

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

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

8.2.3 Update Functions

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

The Numerical Quantities Summary contains the timings for the update functions.

8.2.4 Timing Intervals

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

The Numerical Quantities Summary contains the timings for the throughput test.

8.3 Performance Metrics

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

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

8.4 Reproducibility

A description of the method used to determine the reproducibility of the measurement results must be reported. This must include the performance metrics (QppD, QthD, and QphD) from the reproducibility runs (see Clause 5.4.6).

Two consecutive runs were performed. The following table contains the reproducibility metrics for the system reported in this document.

	QppD@1000GB	QthD@1000GB	QphD@1000GB
Run 1	19137.5	10661.5	14284.1
Run 2	19395.9	10546.3	14302.2

9.0 Clause 6: SUT and Driver Implementation

9.1 Driver

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

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

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

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

9.2 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-D queries, although it is completely capable of processing any arbitrary SQL statement (both DML and DDL).

A separate instance of tpcdbatch in 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 update stream, as well as an input file containing the 17 queries and/or update 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 update function in turn.

For queries, each query is prepared, 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. Update function data is pre-generated with dbgen and pre-split with db2split. UF1 data is further logically split into n portions by appending an extra column to each new ORDERS and LINEITEM record, taking care to ensure that records for the same orderkey remain in the same portion. UF2 data is further physically split into m portions. During the run, when tpcdbatch encounters a call to execute UF1, it first calls a shell script which loads the insert data into staging tables (one LINEITEM staging table and one ORDERS staging table). Then, tpcdbatch forks off n children to insert the new data into the real LINEITEM and ORDERS tables via a subselect from the staging tables, with each child inserting one of n portions. When tpcdbatch encounters a call to execute UF2, it calls a shell script to start m applications on each node, one for each of the m portions of delete data. Each application reads the keys for its portion on its node and performs the deletes using these keys.

10.0 Clause 7: Pricing

10.1 Hardware and Programs Used

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

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

10.2 Five Year Cost of System Configuration

The total 5 year price of the entire configuration must be reported, including: hardware, software, and maintenance charges. Separate component pricing is recommended. The basis of all discounts used must be disclosed.

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

The basis for the discounts is disclosed below:

Mid-Range Service Option (MSRO): The Mid-Range Service Option provides a maintenance service discount to those IBM RS/6000 customers who have installed effective system management controls (problem determination, problem reporting and other processes) and who will commit to an IBM maintenance service coverage period. For this TPC-D report, the IBM configuration qualifies for a seventeen percent discount of the monthly maintenance charges.
Extended Maintenance Option (EMO): The Extended Maintenance Option (EMO) provides reduced maintenance charges for newly purchased RS/6000 machine types. Reduction in charges is achieved through pre-payment and elimination of periodic billing. The discount percentage is based on the term of prepayment which is from 12 to 60 months. EMO may be combined with the Mid-Range Service Option. The EMO discount is applied to the net monthly IBM maintenance charges after the MRSO discount is taken. For this report, the selected prepayment term is one year and yields a seventeen percent discount.

IBM Revenue Discount: IBM Revenue Discount provides reduced pricing on RS/6000 hardware based upon the total list price value. For the RS/6000 SP used in this report, the Revenue Discount is 27%. IBM Software Advantage Discount provides reduced pricing on DB2. The DB2 discount is 33%.

10.3 Availability Dates

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

All hardware and system software used in this test are generally available at the time of publication. The level of DB2 Universal Database Version 5 used will be available October 31, 1998.

11.0 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 audit was conducted by François Raab, who can be contacted at the address below.
Information Paradigm
1373 N. Franklin St.
Colorado Springs
Colorado 80903-2527
(719)-473-7555
email : francois@sizing.com

The auditor's Attestation letter is included at the end of this report.

Appendix A Tunable Parameters

A.1 Database manager configuration parameters

Non-default values are identified with "".*

```
Database Manager Configuration
   Node type = Partitioned Database Server with local and remote clients
 Database manager configuration release level                    = 0x0800
 *CPU speed (millisec/instruction)             (CPUSPEED)        = 1.251712e-06
 *Communications bandwidth                     (COMM_BANDWIDTH)  = 5.000000e+00
 Max number of concurrently active databases   (NUMDB)           = 8
 Transaction processor monitor name            (TP_MON_NAME)     =
 Default charge-back account                   (DFT_ACCOUNT_STR) =
 Java Development Kit 1.1 installation path    (JDK11_PATH)      =
 Diagnostic error capture level                (DIAGLEVEL)       = 3
 *Diagnostic data directory path               (DIAGPATH)        = /tmp/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
   Unit of work                                (DFT_MON_UOW)     = OFF
 SYSADM group name                             (SYSADM_GROUP)    = DBADM
 SYSCTRL group name                            (SYSCTRL_GROUP)   =
 SYSMAINT group name                           (SYSMAINT_GROUP)  =
 Database manager authentication               (AUTHENTICATION)  = SERVER
 Trust all clients                             (TRUST_ALLCLNTS)  = YES
 Trusted client authentication                 (TRUST_CLNTAUTH)  = CLIENT
 Default database path                         (DFTDBPATH)       = /u/tpcd
 Database monitor heap size (4KB)              (MON_HEAP_SZ)     = 48
 UDF shared memory set size (4KB)              (UDF_MEM_SZ)      = 256
 Java Virtual Machine heap size (4KB)          (JAVA_HEAP_SZ)    = 512
 Audit buffer size (4KB)                       (AUDIT_BUF_SZ)    = 0
 Backup buffer default size (4KB)              (BACKBUFSZ)       = 1024
 Restore buffer default size (4KB)             (RESTBUFSZ)       = 1024
 *Sort heap threshold (4KB)                    (SHEAPTHRES)      = 90000
 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
 DRDA services heap size (4KB)                 (DRDA_HEAP_SZ)    = 128
 Priority of agents                            (AGENTPRI)        = SYSTEM
 *Max number of existing agents                (MAXAGENTS)       = 2500
 *Agent pool size                              (NUM_POOLAGENTS)  = 50
 *Initial number of agents in pool             (NUM_INITAGENTS)  = 50
 Max number of coordinating agents             (MAX_COORDAGENTS) = (MAXAGENTS - NUM_INIT AGENTS)
 Max no. of concurrent coordinating agents     (MAXCAGENTS)      = MAX_COORDAGENTS
 Keep DARI process                             (KEEPDARI)        = YES
 Max number of DARI processes                  (MAXDARI)         = MAX_COORDAGENTS
 Index re-creation time                        (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
 TCP/IP Service name                           (SVCENAME)        =
 APPC Transaction program name                 (TPNAME)          =
 IPX/SPX File server name                      (FILESERVER)      =
 IPX/SPX DB2 server object name                (OBJECTNAME)      =
 IPX/SPX Socket number                         (IPX_SOCKET)      = 879E
 Discovery mode                                (DISCOVER)        = SEARCH
 *Discovery communication protocols            (DISCOVER_COMM)   = TCPIP
 Discover server instance                      (DISCOVER_INST)   = ENABLE
 Directory services type                       (DIR_TYPE)        = NONE
```

Directory path name	(DIR_PATH_NAME)	= /.:/subsys/database/
Directory object name	(DIR_OBJ_NAME)	=
Routing information object name	(ROUTE_OBJ_NAME)	=
Default client comm. protocols	(DFT_CLIENT_COMM)	=
*Maximum query degree of parallelism	(MAX_QUERYDEGREE)	= ANY
*Enable intra-partition parallelism	(INTRA_PARALLEL)	= YES
No. of int. communication buffers(4KB)	(FCM_NUM_BUFFERS)	= 4096
*Number of FCM request blocks	(FCM_NUM_RQB)	= 15360
*Number of FCM connection entries	(FCM_NUM_CONNECT)	= 15000
Number of FCM message anchors	(FCM_NUM_ANCHORS)	= (FCM_NUM_RQB * 0.75)
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.2 Database configuration parameters

Non-default values are identified with "".*

Database Configuration for Database TPCD1TB

Database configuration release level		= 0x0800
Database release level		= 0x0800
Database territory		= en_US
Database code page		= 819
Database code set		= ISO8859-1
Database country code		= 1
Directory object name	(DIR_OBJ_NAME	=
Discovery support for this database	(DISCOVER_DB)	= ENABLE
*Degree of parallelism	(DFT_DEGREE)	= 4
*Default query optimization class	(DFT_QUERYOPT)	= 7
Continue upon arithmetic exceptions	(DFT_SQLMATHWARN)	= NO
Number of frequent values retained	(NUM_FREQVALUES)	= 10
Number of quantiles retained	(NUM_QUANTILES)	= 20
Backup pending		= NO
Database is consistent		= YES
Rollforward pending		= NO
Restore pending		= NO
Multi-page file allocation enabled		= NO
Log retain for recovery status		= NO
User exit for logging status		= NO
Datalink Access Token Expiry Interval (sec)	(DL_EXPINT)	= 1
Datalink Number of Copies	(DL_NUM_COPIES)	= 1
Datalink Number of Backups	(DL_NUM_BACKUP)	= 1
Datalink Time after Drop (days)	(DL_TIME_DROP)	= 1
*Database heap (4KB)	(DBHEAP)	= 2048
Catalog cache size (4KB)	(CATALOGCACHE_SZ)	= 64
*Log buffer size (4KB)	(LOGBUFSZ)	= 512
*Utilities heap size (4KB)	(UTIL_HEAP_SZ)	= 8000
*Buffer pool size (4KB)	(BUFFPAGE)	= 275000
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)	= 8192
*Max appl. control heap size (4KB)	(APP_CTL_HEAP_SZ)	= 512
*Sort list heap (4KB)	(SORTHEAP)	= 15000
*SQL statement heap (4KB)	(STMTHEAP)	= 2048
*Default application heap (4KB)	(APPLHEAPSZ)	= 64
*Package cache size (4KB)	(PCKCACHESZ)	= 1024
Statistics heap size (4KB)	(STAT_HEAP_SZ)	= 4384
Interval for checking deadlock (ms)	(DLCHKTIME)	= 10000
*Percent. of lock lists per application	(MAXLOCKS)	= 13
Lock timeout (sec)	(LOCKTIMEOUT)	= -1
Changed pages threshold	(CHNGPGS_THRESH)	= 60
*Number of asynchronous page cleaners	(NUM_IOCLEANERS)	= 8
*Number of I/O servers	(NUM_IOSERVERS)	= 24
Index sort flag	(INDEXSORT)	= YES
Sequential detect flag	(SEQDETECT)	= YES
*Default prefetch size (4KB)	(DFT_PREFETCH_SZ)	= 128
Default number of containers		= 1
Default tablespace extentsize (4KB)	(DFT_EXTENT_SZ)	= 32
*Max number of active applications	(MAXAPPLS)	= 2400
Average number of active applications	(AVG_APPLS)	= 1

*Max DB files open per application	(MAXFILOP)	= 1000
*Log file size (4KB)	(LOGFILSIZ)	= 3500
*Number of primary log files	(LOGPRIMARY)	= 50
*Number of secondary log files	(LOGSECOND)	= 5
Changed path to log files	(NEWLOGPATH)	=
Path to log files	= /database/tpcd/NODE0000/SQL00001/SQLOGDIR/	
Next active log file		=
First active log file		=
Group commit count	(MINCOMMIT)	= 1
Percent log file reclaimed before soft chckpt	(SOFTMAX)	= 100
Log retain for recovery enabled	(LOGRETAIN)	= OFF
User exit for logging enabled	(USEREXIT)	= OFF
Auto restart enabled	(AUTORESTART)	= ON
Index re-creation time	(INDEXREC)	= SYSTEM (RESTART)
Default number of loadrec sessions	(DFT_LOADREC_SES)	= 1
Recovery history retention (days)	(REC_HIS_RETENTN)	= 366
ADSM management class	(ADSM_MGMTCLASS)	=
ADSM node name	(ADSM_NODENAME)	=
ADSM owner	(ADSM_OWNER)	=
ADSM password	(ADSM_PASSWORD)	=

A.3 DB2 Registry Variables

DB2_IORETRY_LEVEL	= '1'
DB2_PRED_FAC	= 'Y'
DB2_SORT_BUFF	= 'Y'
DB2_MGJN_WITH_SORT	= 'Y'
DB2ENABLECLUSTER	= 'yes'
DB2_LIKE_VARCHAR	= 'Y'
DB2_CORRELATED_PREDICATES	= 'Y'
DB2_VECTOR	= 'Y'
DB2MEMMAXFREE	= '8000000'
DB2MEMDISCLAIM	= 'yes'
DB2_MMAP_WRITE	= 'no'
DB2_MMAP_READ	= 'no'
DB2_RR_TO_RS	='Y'
DB2OPTIONS	= '-t -v +c'

A.4 AIX Parameters

Non-default values are identified with "".*

keylock	normal	State of system keylock at boot time	False
maxbuf	20	Maximum number of pages in block I/O BUFFER CACHE	True
maxmbuf	0	Maximum Kbytes of real memory allowed for MBUFS	True
*maxuproc	2500	Maximum number of PROCESSES allowed per user	True
autorestart	false	Automatically REBOOT system after a crash	True
iostat	true	Continuously maintain DISK I/O history	True
realmem	3137536	Amount of usable physical memory in Kbytes	False
conslogin	enable	System Console Login	False
fwversion	IBM,L98091	Firmware version and revision levels	False
maxpout	0	HIGH water mark for pending write I/Os per file	True
minpout	0	LOW water mark for pending write I/Os per file	True
fullcore	false	Enable full CORE dump	True
rtasversion 1	Open Firmware	RTAS version	False
modelname	IBM,9076-WCN	Machine name	False
boottype	disk	N/A	False

vmtune: current values:

-p	-P	-r	-R	-f	-F	-N	-W
minperm	maxperm	minpgahead	maxpgahead	minfree	maxfree	pd_npages	maxrandwrt
78433	313734	2	*128	*1024	1024	524288	0

-M	-w	-k	-c	-b	-B	-u
maxpin	npswarn	npskill	numclust	numfsbufs	hd_pbuf_cnt	lvm_bufcnt
627468	2000	512	1	93	625	9

A.5 Network Parameters

Non-default values are identified with "".*

*thewall	= 16384
*sb_max	= 3000000
somaxconn	= 1024
clean_partial_conns	= 0
net_malloc_police	= 0
rto_low	= 1
rto_high	= 64
rto_limit	= 7
rto_length	= 26
arptab_bsiz	= 7
arptab_nb	= 25
tcp_ndebug	= 100
ifsize	= 8
arpqsize	= 1
route_expire	= 0
strmsgsz	= 0
strctlsz	= 1024
nstrpush	= 8
strthresh	= 85
psetimers	= 20
psebufcalls	= 20
strturncnt	= 15
pseintrstack	= 12288
lowthresh	= 90
medthresh	= 95
psecache	= 1
subnetsarelocal	= 1
maxttl	= 255
ipfragttl	= 60
ipsendredirects	= 1
ipforwarding	= 0
udp_ttl	= 30
tcp_ttl	= 60
arpt_killc	= 20
*tcp_sendspace	= 221184
*tcp_recvspace	= 221184
*udp_sendspace	= 65536
*udp_recvspace	= 655360
rfc1122addrchk	= 0
nonlocsrcroute	= 1
tcp_keepintvl	= 150
tcp_keepidle	= 14400
bcastping	= 0
udpcksum	= 1
tcp_mssdflt	= 4044
icmpaddressmask	= 0
tcp_keepinit	= 150
ie5_old_multicast_mapping	= 0
rfc1323	= 1
pmtu_default_age	= 10
pmtu_rediscover_interval	= 30
udp_pmtu_discover	= 0
tcp_pmtu_discover	= 0
ipqmaxlen	= 100
directed_broadcast	= 1
ipignoreredirects	= 0
ipsrcroutesend	= 1
ipsrcrouterecv	= 0
ipsrcrouteforward	= 1
portcheck	= 0
udpchecksum	= 1
nfs_socketsize	= 2000000
nfs_tcp_socketsize	= 442240
nfs_setattr_error	= 0
nfs_gather_threshold	= 4096
nfs_repeat_messages	= 0
nfs_udp_duplicate_cache_size	= 0
nfs_tcp_duplicate_cache_size	= 0
nfs_server_base_priority	= 0
nfs_dynamic_retrans	= 1
nfs_iopace_pages	= 0
nfs_max_connections	= 0
nfs_max_threads	= 5
nfs_use_reserved_ports	= 0
nfs_device_specific_bufs	= 1
nfs_server_clread	= 1
nfs_max_write_size	= 0
nfs_max_read_size	= 0

A.6 Switch parameters

Non-default values are identified with "".*

kernel_memory	0xf1000000
TB3MX kernel memory address	False
user_memory	0xf2000000
TB3MX user memory address	False
int_priority	3
Interrupt priority	False
int_level	14
Bus interrupt level	False
*spoolsize	0x1000000
Size of IP send buffer	True
*rpoolsize	0x1000000
Size of IP receive buffer	True
adapter_status	css_ready
Configuration status	False

Appendix B Database Build Scripts

B.1 Node configuration files

B.1.1 db2nodes.cfg

```
00 db2sn01 0 db2sn01
01 db2sn02 0 db2sn02
02 db2sn03 0 db2sn03
03 db2sn04 0 db2sn04
04 db2sn05 0 db2sn05
05 db2sn06 0 db2sn06
06 db2sn07 0 db2sn07
07 db2sn08 0 db2sn08
08 db2sn09 0 db2sn09
09 db2sn10 0 db2sn10
10 db2sn11 0 db2sn11
11 db2sn12 0 db2sn12
12 db2sn13 0 db2sn13
13 db2sn14 0 db2sn14
14 db2sn15 0 db2sn15
15 db2sn16 0 db2sn16
16 db2sn17 0 db2sn17
17 db2sn18 0 db2sn18
18 db2sn19 0 db2sn19
19 db2sn20 0 db2sn20
20 db2sn21 0 db2sn21
21 db2sn22 0 db2sn22
22 db2sn23 0 db2sn23
23 db2sn24 0 db2sn24
24 db2sn25 0 db2sn25
25 db2sn26 0 db2sn26
26 db2sn27 0 db2sn27
27 db2sn28 0 db2sn28
28 db2sn29 0 db2sn29
29 db2sn30 0 db2sn30
30 db2sn31 0 db2sn31
31 db2sn32 0 db2sn32
32 db2sn33 0 db2sn33
33 db2sn34 0 db2sn34
34 db2sn35 0 db2sn35
35 db2sn36 0 db2sn36
36 db2sn37 0 db2sn37
37 db2sn38 0 db2sn38
38 db2sn39 0 db2sn39
39 db2sn40 0 db2sn40
40 db2sn41 0 db2sn41
41 db2sn42 0 db2sn42
42 db2sn43 0 db2sn43
43 db2sn44 0 db2sn44
44 db2sn45 0 db2sn45
45 db2sn46 0 db2sn46
46 db2sn47 0 db2sn47
47 db2sn48 0 db2sn48
```

B.1.2 db2nodes.qual.cfg

```
40 db2sn41 0 db2sn41
41 db2sn42 0 db2sn42
42 db2sn43 0 db2sn43
43 db2sn44 0 db2sn44
```

B.2 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_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" )
   {
      $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_DDL"}) <= 0)
{
   $ENV{"TPCD_DDL"} = "dss.ddl";
}
if (length($ENV{"TPCD_TBSP_DDL"}) <= 0)
{
   $ENV{"TPCD_TBSP_DDL"} = "dss.tbsp.ddl";
}
if (length($ENV{"TPCD_INDEXDDL"}) <= 0)
{
   $ENV{"TPCD_INDEXDDL"} = "dss.index";
}
if (length($ENV{"TPCD_RUNSTATS"}) <= 0)
{
   $ENV{"TPCD_RUNSTATS"} = "dss.runstats";
}
if ( ($ENV{"TPCD_INPUT"}) eq "NULL" )
{
   if (length($ENV{"TPCD_DBGEN"}) <= 0)
   {
      die "Must set TPCD_DBGEN if pregenerated flatfiles are not
provided (TPC
D_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_LOG_DIR"}) <= 0)
{
   $ENV{"TPCD_LOG_DIR"} = "NULL";
}
if (length($ENV{"TPCD_CONFIGFILE"}) <= 0)
{
   $ENV{"TPCD_CONFIGFILE"} = "dss.dbconfig";
}
if (length($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
";
}



#set up local variables
$product=$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"};
$tbspddl=$ENV{"TPCD_TBSP_DDL"};
$indexddl=$ENV{"TPCD_INDEXDDL"};
$extraindex=$ENV{"TPCD_EXTRAINDEX"};
$runstats=$ENV{"TPCD_RUNSTATS"};
$dbgen=$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"};
if ( $fparse eq "yes" )
{
   $fastparse="FASTPARSE";
}
else
{
   $fastparse=" ";
}
$backupdir=$ENV{"TPCD_BACKUP_DIR"};
```

```
$log=$ENV{"TPCD_LOG"};
$logDir=$ENV{"TPCD_LOG_DIR"};
$machine=$ENV{"TPCD_MACHINE"};
$configfile=$ENV{"TPCD_CONFIGFILE"};
$loadconfigfile=$ENV{"TPCD_LOAD_CONFIGFILE"};
$smpdegree=$ENV{"TPCD_SMPDEGREE"};
$agentpri=$ENV{"TPCD_AGENTPRI"};
$activate=$ENV{"TPCD_ACTIVATE"};
$RealAudit=$ENV{"TPCD_AUDIT"};
$loadscript=$ENV{"TPCD_LOAD_SCRIPT"};

if ( $RealAudit eq "yes" )
{
   # need some extra parameters for some of the setup
   if (length($ENV{"TPCD_AUDIT_DIR"}) <= 0)
   {
      die "TPCD_AUDIT_DIR environment variable not set\n";
   }
   $auditDir=$ENV{"TPCD_AUDIT_DIR"};
   $user=$ENV{"USER"};
}

# set up override of some parameters to override for qualification database
if ( $qual eq "QUAL" )
{
   $loadscript=$ENV{"TPCD_LOAD_SCRIPT_QUAL"};
   if ( length($ENV{"TPCD_QUAL_DBNAME"}) <= 0 )
   {
      die "TPCD_QUAL_DBNAME environment variable not set\n";
   }
   $dbname=$ENV{"TPCD_QUAL_DBNAME"};
   $sf=0.100;
   if ( length($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_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 = 10000;
     $sortheap =  8000;
     $sheapthres = 20000;
     $util_heap_sz = 10000;
     $ioservers = 10;
  }
  elsif ( $machine eq "big" )
  {
     $buffpage = 30000;
     $sortheap = 20000;
     $sheapthres = 50000;
     $util_heap_sz = 30000;
     $ioservers = 20;
  }
  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 = 81000;
     $ioclnrs = 4;
     $chngpgs = 30;
     $ioservers = 21;
     $util_heap_sz = 30000;
  }
}
# echo parameter settings to acknowledge what is being built
print "Building a TPC-D $sf GB database on $dbpath with: \n";
print "   Mode = $mode \n";
print "   Tablespace ddl in $ddlpath${delim}$tbspddl \n";
print "   Table ddl in $ddlpath${delim}$ddl \n";
print "   Index ddl in              $ddlpath${delim}$indexddl\n";
if ( $extraindex ne "no" )
{
  print "             and             $ddlpath${delim}$extraindex\n";
}
print "   Runstats in               $ddlpath${delim}$runstats \n";
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 ( $loadstats eq "yes" )
{
  if ( $earlyindex eq "yes" )
  {
    print "   Statistics for tables and indexes gathered during load\n";
  }
  else
  {
    print "   Statistics for tables and indexes gathered after load\n";
  }
}
else
{
  print "   Statistics for tables and indexes gathered after load\n";
}
if ( $loadconfigfile ne "NULL" )
{
  print "   Configuration parameters for LOAD taken from
$ddlpath${delim}$loadco
nfigfile\n";
}
if ( $configfile ne "NULL" )
{
  print "   Configuration parameters taken from $configfile\n";
}
else
{
  print "   Configuration paramters taken from
$ddlpath${delim}dss.dbconfig${sfR
eal}GB\n";
  $configfile="dss.dbconfig${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 (($loadconfigfile eq "") || ($loadconfigfile eq "NULL"))
{
  print "   Machine size set to $machine so the following configuration\n";
  print "   parameters are used for load, create index and runstats: \n";
  print "      BUFFPAGE = $buffpage \n";
  print "      SORTHEAP = $sortheap \n";
  print "      SHEAPTHRES = $sheapthres\n";
  print "      NUM_IOSERVERS = $ioservers\n";
  print "      NUM_IOCLEANERS = $ioclnrs\n";
  print "      CHNGPGS_THRESH = $chngpgs\n";
  print "      UTIL_HEAP_SZ = $util_heap_sz\n";
  print "      Degree of parallelism (dft_degree and max_querydegree) set to
$sm
pdegree\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";
}
else
{
  if (( $mode eq "mln" ) || ( $mode eq "mpp"))
  {
    die "   Loading script must be specified for mln or mpp invocations\n";
  }
}
print "   Degree of parallelism (dft_degree and max_querydegree) set to
$smpdegr
ee\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";
}

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

# set db2options so all usages work
$rc=system("db2set DB2OPTIONS=\"-t -v +c\" -i ");
if ( $platform eq "nt" )
{
   $rc = system("db2set DB2NTNOCACHE=ON");
}

#
# goto createUFfiles;

# labels available:
#   createdb
#   aftercreatedb
#   createtbsp
#   createUFfiles
#   createtbl
#   startload
#   createndx
#   getstats
#   addRI
#   setruncfg
#   setupTblFns

# starting db2
print "Starting DB2 ...\n";
$rc=system("db2start");
if ( $rc != 0 )
{
   die "failure during db2start rc = $rc \n";
}



createdb:
# create the database
&outtime("*** Starting to create the database");

&dodb_noconn("db2 \"create database $dbname on $dbpath collate using
identity wi
th 'TPC-D $sf GB'\"",$once);
```

```
aftercreatedb:
# reset the db and dbm configuration before we start
# and also increase MAXAPPLS at this node
&dodb_noconn("db2 reset database configuration for $dbname  $sep  \
            db2 update db cfg for $dbname using MAXAPPLS 80 ",$all_ln);
&dodb_noconn("db2 reset database manager configuration $sep       \
            db2 update dbm cfg using diagpath /tmp/db2dump ",$once);
&dodb_conn($dbname,"db2 alter bufferpool ibmdefaultbp size -1 $sep   \
            db2 grant connect on database to public $sep           \
            db2 grant dbadm on database to $dbname $sep            \
            db2 commit",$once);

if (($loadconfigfile eq "") || ($loadconfigfile eq "NULL"))
{
 &dodb_conn($dbname,
          "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);
}

# update the log information first
# set up the log directory before we do any index creation
if ( $logDir ne "NULL" )
{
   &dodb_noconn("db2 update database configuration for $dbname using
newlogpath
 $logDir",$all_ln);
}
&dodb_noconn("db2 update db cfg for $dbname using logprimary 50
logsecond 5 logf
ilsiz 3500 logbufsz 512", $all_ln);

# if logging is enabled, we must take a backup of the database
if ( $log eq "yes" )
{
  &dodb_conn($dbname,"db2 update database configuration for $dbname
using LOGRET
AIN yes",$all_ln);
  print "\n NOTE:  DO NOT RESET THE DATABASE
CONFIGURATION or you will lose logr
etain\n";
  &outtime("*** Starting the backup");
#need to test parallel specific
  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
$backupdi
r 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
$backupdir",$once);
  }
  &outtime("*** Finished the backup");
}


createtbsp:
# create the tables
&outtime("*** Ready to start creating the tablespaces");
&dodb2file($dbname,"$ddlpath${delim}$tbspddl",$once);


createUFfiles:
&dodb2file($dbname,"$ddlpath${delim}createUFtbls",$once);


createtbl:
&outtime("*** Start Load Timing now - starting to create tables");
&dodb2file($dbname,"$ddlpath${delim}$ddl",$once);

# update the locksize on the non-updated tables to be table level locking
# and update the table for appendmode
&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 \
           db2 alter table tpcd.lineitem append on $sep \
           db2 alter table tpcd.orders append on",
          $once);

#if ( $mode eq "mpp" )
#{
#   #need parallel specific
#    print "need to figure parallel specific creation of tmp\n";
#}
#mkdir("${delim}tmp${delim}$instance",0777);

# setup the the load configuration
&outtime("*** Setting LOAD configuration.");
if (($loadconfigfile eq "") || ($loadconfigfile eq "NULL"))
{
 &dodb_conn($dbname,
          "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($dbname,"$ddlpath${delim}$loadconfigfile.dbm",$once);
   &dodb2file($dbname,"$ddlpath${delim}$loadconfigfile",$all_ln);
}
&dodb_noconn("db2 terminate",$once);
$rc=system("db2stop");
$rc=system("db2_kill");
```

```
$rc=system("db2start");
if ( $rc != 0 )
{
   die "failure during db2start rc = $rc \n";
}
&dodb_noconn("activate database $dbname",$once);

startload:
# set up the log directory before we do any index creation
if ( $logDir ne "NULL" )
{
   &dodb_noconn("db2 update database configuration for $dbname using
newlogpath
 $logDir",$all_ln);
}

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

  if ( $extraindex ne "no" )
  {
     # use this additional file for indexes
    &dodb2file($dbname,"$ddlpath${delim}$extraindex",$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 ne "NULL") && ( $loadscript ne "" ))
  {
   &dodb2file($dbname,"$ddlpath${delim}$loadscript",$once);
  }
  else
  {
    $rc = system("perl genloaduni $qual");
    if ( $rc != 0 )
    {
      die "genloaduni failed rc = $rc\n";
    }
  }
}
elsif (( $mode eq "mln" ) || ( $mode eq "mpp" ))
{
  &outtime("*** Starting the load");
    # call the appropriate dbgen/split/(sort)/load for mln/mpp
  $rc = system("$ddlpath${delim}$loadscript $sf");
  if ( $rc != 0 )
  {
     die "doload for $dbname failed rc = $rc\n";
  }
}
else
{
   die "TPCD_MODE not set to one of uni, smp, mln or mpp\n";
}


createndx:
# if indexes haven't been created, do so now
if ( $earlyindex ne "yes" )
{
  &outtime("*** Create index started");
  &dodb2file($dbname,"$ddlpath${delim}$indexddl",$once);
  if ( $extraindex ne "no" )
  {
     # use this additional file for indexes
    &dodb2file($dbname,"$ddlpath${delim}$extraindex",$once);
  }
  &outtime("*** Create index completed");
}


getstats:
# if statistics not gathered on the load, run runstats  (we have to run the
# stats at the same time whether it be both during load, or after load)
if (( $loadstats eq "no" ) || ( $earlyindex eq "no"))
{
  # if loadstats not gathered, then index stats not gathered either.
  &outtime("*** Runstats started");

  # start all but lineitem in background on node 1
  $rc=system("db2_all \'}]<<+001< db2 -tvf
\"$ddlpath${delim}$runstats.notline\"
 > runstats.node1.out \' & ");

  # start lineitem in foreground; it should take much longer than all other
tabl
es
  &dodb2file($dbname,"$ddlpath${delim}$runstats.line",$once);
  &outtime("*** Runstats completed");
}


addRI:
# add constraints
&outtime("*** Adding RI contraints started");
&dodb2file($dbname, "$ddlpath/dss.ri3", $once);
&outtime("*** Adding RI contraints completed");


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

if ( ($mode eq "uni") || ($mode eq "smp"))
{
   &dodb2file(${dbname},"${ddlpath}${delim}${configfile}",$once);
}
elsif ( $mode eq "mln" || $mode eq "mpp" )
{
   &dodb2file($dbname,"$ddlpath${delim}$configfile.dbm",$once);
   &dodb2file($dbname,"$ddlpath${delim}$configfile",$all_ln);
}


if ( $agentpri ne "NULL" )
{
   &dodb_noconn("db2 update dbm cfg using AGENTPRI
$agentpri",$once);
}
# stop and restart the database to get config parms recognized
&dodb_noconn("deactivate database $dbname",$once);
$rc=system("db2stop");
$rc=system("db2_kill");
$rc=system("db2start");
if ( $rc != 0 )
{
   die "failure during db2start rc = $rc \n";
```

```
}
&dodb_noconn("activate database $dbname",$once);

&outtime("*** Set Configuration completed");


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${delim}$dbname.$user.update.pair.num");
   # &rm("$auditDir${delim}tools${delim}tpcd.runsetup");
   system("rm /home/tpcd/tpcd/tools/tpcd.runsetup");
   system("rm /home/tpcd/tpcd/TPCD1TB.tpcd.update.pair.num" );
   system("perl setupDir");
   system("perl setupRun");
   # before we stop the database for the final time
   # if we are in the real audit mode then compile and bind tpcdbatch, and
   # run dbtables and dbcheck before we print out the final notice that
   # we are ready to run the performance tests
   # if we are building the qualification database then we will bind to both
   # the dbname database and the qualification database
   $rc = system("perl buildtpcdbatch $qual");
   if ( $rc != 0 )
   {
      die "buildtpcdbatch failed rc=$rc\n";
   }
   if ( $qual eq "QUAL" )
   {
      $verifyType="q";
   }
   else
   {
      $verifyType="t";
   }
# dbcheck no longer needed
#    system("perl checkdb $verifyType");
   system("perl tablesdb $verifyType");
   &dodb2file($dbname,"/home/tpcd/tpcd/tools/first10rows.sql",$once);
}

# db2stop
&dodb_noconn("deactivate database $dbname",$once);
$rc=system("db2stop");
$rc=system("db2_kill");

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

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

# finished creating the database
&outtime("*** Finished creating the database");
1;
```

B.3 tpcd.setup

```
# NOTE: ALL variable defitions must have a comment at the end
TPCD_PLATFORM=aix                 # aix, nt, ....
TPCD_DBNAME=TPCD1TB               # name to create database under
TPCD_AUDIT_DIR=/u/tpcd/tpcd       # top level directory of tar file for
                                  # all the tpcd scripts
TPCD_PRODUCT=v5                   # v5 or pe
                                  # Use pe if you really are using pe v1.2!
                                  # but I won't guarantee that it will work!
TPCD_MODE=mpp                     # uni/smp/mln/mpp
TPCD_PHYS_NODE=48                 # number of physical nodes
TPCD_LN_PER_PN=1                  # number of logical nodes per
                                    physical node
TPCD_SF=1000                      # size of the database (1=1GB,...) to
                                  # get test size databases use:
                                  #     0.012 = 12MB
                                  #     0.1   = 100MB
TPCD_BUILD_DATABASE=NULL  # whether to build a new database
                                    # NOT CURRENTLY USED
TPCD_DBPATH=/database       # path for database (defaults to home)
TPCD_DDLPATH=/u/tpcd/tpcd/1TBstuff     # path for all ddl files
TPCD_TBSP_DDL=dss.ddl1TB.tbsp.pok      # ddl file for tablespaces
TPCD_DDL=dss.ddl1TB.tbl.pok            # ddl file for tables
TPCD_QUAL_TBSP_DDL=dss.ddl1TB.tbsp.pok.qual
                                  # ddl file for tablespaces for qual
TPCD_QUAL_DDL=dss.ddl1TB.tbl.pok.qual
                                  # ddl file for qualification database
                                  # tablespaces and tables should be identical
                                  # to regular ddl except container names
TPCD_INDEXDDL=dss.index.all.include    # ddl file for indexes
TPCD_EXTRAINDEX=no       # use this additional file when creating
                                  # indexes
TPCD_RUNSTATS=dss.runstats              # ddl file for runstats
TPCD_DBGEN=/u/tpcd/tpcd/appendix/dbgen
                                  # path name to data generation code
TPCD_INPUT=/load                  # NULL - use dbgen generated data  OR
                                  # path name -  to the pre-generated
                                  #              flat files
                        # /gwl/dss/12MB - path for pregenerated 12MB
                        # /gwl/dss/100MB - path for pregen'd 100MB
TPCD_QUAL_INPUT=/qflat   # NULL - use dbgen generated data  OR
                        # path name -  to the pre-generated
                        #              flat files
TPCD_TAILOR_DIR=/u/tpcd/tpcd/tailor
                        # path name for the directory that
                        # contains any tailoring specific files
                        # or code....eg split config files

TPCD_EARLYINDEX=no              # create indexes before the load
                                  # LOAD specific parameters follow:
TPCD_LOAD_CONFIGFILE=dss.ldconfig1TB.pok
                        # config file with specific load config
                        # parms set to NULL if use defaults
TPCD_TEMP=              # path for LOAD temp files
                        # defaults to /u/<instance>/sqllib/tmp
TPCD_SORTBUF=4096                 # sortbuf size for LOAD
TPCD_LOAD_PARALLELISM=0
                        # degree of parallelism to use on load
                        # 0 = use the "intelligent default" that
                        # the load will chose at run time
TPCD_LOADSTATS=no                  # gather statistics during load
                        # ignored if EARLYINDEX is not set
                        # due to runstats limitation
TPCD_COPY_DIR=/dev/null
                        # directory where copy image is created
                        # on load command CURRENTLY UNUSED
TPCD_FASTPARSE=yes                # use fastparse on load

                        # Backup specific parameters follow:
```

```
TPCD_BACKUP_DIR=/u/tpcd/backupdir
                        # directory where backup files are placed

TPCD_LOG_DIR=NULL     # directory where log files stored..
                        # NULL leaves them in the dbpath
TPCD_LOG_QUAL_DIR=/qualdb
                        # directory where qual log files stored
                        # NULL leaves them in the dbpath
TPCD_LOG=no                     # yes/no - whether to turn
LOG_RETAIN on
                        # i.e. are backups needed to be taken
                        # CONFIG specific parameters
TPCD_CONFIGFILE=dss.dbconfig1TB.pok
                        # name of configuration file in ddl path
TPCD_DBM_CONFIG=NULL
                        #  not used yet  Will be used when
                        #   dbm cfg set up separately from db cfg
TPCD_QUALCONFIGFILE=dss.dbconfig1TB.pok.qual
                        # name of configuration file in ddl path
                        # for qualification database
TPCD_DBM_QUALCONFIG=NULL
                        #  not used yet  Will be used when
                        #   dbm cfg set up separately from db cfg
TPCD_MACHINE=big
                        # big/medium/small  size of machine used to
                        # determine buffpage, sortheap,sheapthres
                        # and ioservers parms for load, create
                        # index and runstats
TPCD_SMPDEGREE=4     # 1...# of degrees of parallelism to run
                        # with
TPCD_AGENTPRI=NULL
                        # set agentpri to this value (default
                        # is SYSTEM)

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

TPCD_QUERY_TEMPLATE_DIR=mln
                        # subdirectory in AUDIT_DIR/queries
                        # to use as the source of the query
                        # templates.  Currently there are
                        # v2 ones and pe ones.  You can make
                        # your own directory following the same
                        # form as in the v2 directory using
                        # any variant you wish
TPCD_QUAL_DBNAME=TPCDQUAL
                       # name of qualification database
TPCD_NUMSTREAM=7
                        # number of streams for the throughput test
TPCD_FLATFILES=/uftbsp
                       # where to generate flat files
                       # for update functions
TPCD_SPLIT_UPDATES=48
                       # number of chunks to split the update
                       # function into.
TPCD_CONCURRENT_INSERTS=48
                        # number of insert chunks that are run
                        # concurrently.  This number should be
                        # evenly divisible by TPCD_SPLIT_UPDATES
TPCD_CONCURRENT_INSERTS_LOAD=8
                        # number of insert chunks that are loaded
                        # concurrently.  This number should be
                        # evenly divisible by TPCD_SPLIT_UPDATES
                        # this controls the load portion of the
                        # insert routine
TPCD_SPLIT_DELETES=48
                        # number of portions to split the delete
                        # function into.
                        # this variable is only valid in UNI/SMP
                        # mode.  It is ignored in MLN/MPP mode
TPCD_GEN_UPDATEPAIRS=24
                        # number of pairs of update function data
                        # to generate
                        # if 0 the update data generation and
                        # setup will not be done.  use this if
                        # you don't want to run the update
                        # functions   (Update functions not
                        # fully tested in new env't yet)
TPCD_GENERATE_SEED_FILE=yes           # yes/no
                        # yes - generate a seed file base on
                        #       year/month/day (for audited runs)
                        # no  - use dbgen's default seeds
TPCD_RUN_ON_MULTIPLE_NODES=NULL
                        # pe only - will we be running each query
                        # stream of throughput starting at
                        # different nodes or from same node
TPCD_STATS_INTERVAL=30
                        # timing interval for vmstats/iostats
TPCD_GATHER_STATS=off
                        # on/off   - only implement for AIX yet
                        # on = gather statistics around power
                        #      test run  (vmstat,iostat,netstat)
                        # off = no stats gathered during power run

TPCD_UFTEMP=UFTEMP
                        # base name of tablespace(s) where the
                        # staging tables for the update functions

                        # are created
                        # this name will be used as the
                        # basename for the tablespaces...eg
                        # UFTEMP1 UFTEMP2 ....
TPCD_HAVECOMPILER=yes                 # rebuild tpcdbatch executable
                                      # yes/no
TPCD_SLEEP=5                          # ?
TPCD_INLISTMAX=708
                        # max num of keys to delete at a time
                        # for UF2, use "default" for default.
TPCD_LOAD_SCRIPT=doload.ksh
                        # script to start for loading tables
                        # in ddl directory under mln/mpp
TPCD_LOAD_SCRIPT_QUAL=doqload.ksh
                        # script to for loading tables in ddl
                        # directory under mln/mpp for QUAL db
                        #  acid test specific information
TPCD_DB2LOG=/u/tpcd/sqllib/db2dump
                        # directory where the db2diag.log can
                        # be found for the durability tests
#TPCD_UPDATE_IMPORT=TF
                       # use table function for UF1 LINEITEM and ORDERS
TPCD_UPDATE_IMPORT=FALSE
                       # use staging tables for UF1 LINEITEM and ORDERS
```

B.4 dss.ddl1TB.tbsp.pok

```
-- SMS temp tablespace
CREATE TEMPORARY TABLESPACE TPCDTEMP PAGESIZE 4096
MANAGED BY SYSTEM USING(
        '/temp2/temp',
        '/temp3/temp',
        '/temp4/temp',
        '/temp5/temp',
        '/temp6/temp',
```

```
            '/temp7/temp',
            '/temp8/temp',
            '/temp9/temp'
        )
        EXTENTSIZE 32 PREFETCHSIZE 512;
COMMIT WORK;
DROP TABLESPACE TEMPSPACE1;

-- DMS data and index tablespace
-- each node has 32 disks (2-9,14-21,22-29,34-41)
-- each disk has a LV of 230PPs (8MB PPs) for data and index
-- pages per container = ( 230 PP ) * ( 8MB/PP ) * (256 pages/MB )
--                     = 471040 pages
-- pages for data42 = ( 100 PP ) * ( 8MB/PP ) * (256 pages/MB )
--                  = 204800 pages
CREATE REGULAR TABLESPACE TPCDALL PAGESIZE 4096
MANAGED BY DATABASE USING(
            DEVICE '/dev/rdata2' 471040,
            DEVICE '/dev/rdata3' 471040,
            DEVICE '/dev/rdata4' 471040,
            DEVICE '/dev/rdata5' 471040,
            DEVICE '/dev/rdata6' 471040,
            DEVICE '/dev/rdata7' 471040,
            DEVICE '/dev/rdata8' 471040,
            DEVICE '/dev/rdata9' 471040,
            DEVICE '/dev/rdata14' 471040,
            DEVICE '/dev/rdata15' 471040,
            DEVICE '/dev/rdata16' 471040,
            DEVICE '/dev/rdata17' 471040,
            DEVICE '/dev/rdata18' 471040,
            DEVICE '/dev/rdata19' 471040,
            DEVICE '/dev/rdata20' 471040,
            DEVICE '/dev/rdata21' 471040,
            DEVICE '/dev/rdata22' 471040,
            DEVICE '/dev/rdata23' 471040,
            DEVICE '/dev/rdata24' 471040,
            DEVICE '/dev/rdata25' 471040,
            DEVICE '/dev/rdata26' 471040,
            DEVICE '/dev/rdata27' 471040,
            DEVICE '/dev/rdata28' 471040,
            DEVICE '/dev/rdata29' 471040,
            DEVICE '/dev/rdata34' 471040,
            DEVICE '/dev/rdata35' 471040,
            DEVICE '/dev/rdata36' 471040,
            DEVICE '/dev/rdata37' 471040,
            DEVICE '/dev/rdata38' 471040,
            DEVICE '/dev/rdata39' 471040,
            DEVICE '/dev/rdata40' 471040,
            DEVICE '/dev/rdata41' 471040,
            DEVICE '/dev/rdata42' 204800
        )
        EXTENTSIZE 32 PREFETCHSIZE 512;
COMMIT WORK;

CREATE NODEGROUP NODE_ONE ON NODE(1);
COMMIT WORK;

CREATE REGULAR TABLESPACE TS_NODE1 IN NODEGROUP
NODE_ONE
  PAGESIZE 4096 MANAGED BY DATABASE
  USING (DEVICE '/dev/rdmsone' 2048) ON NODES (1)
  EXTENTSIZE 32 PREFETCHSIZE 32;
COMMIT WORK;

alter bufferpool ibmdefaultbp size -1;
commit work;

-- create tbspaces for updates
--
create tablespace UFTEMP1 managed by system using
('/uftbsp/uf_chunk1')
  extentsize 32 prefetchsize 32;
commit;
```

B.5 createUFtbls

```
CREATE TABLE TPCDTEMP.ORDERS_NEW ( APP_ID INTEGER
NOT NULL, O_ORDERKEY BIGINT NO
T 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) PARTITIONING KEY (O_ORDERKEY) IN UFTEMP1;

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 I
NTEGER NOT NULL, L_QUANTITY FLOAT NOT NULL,
L_EXTENDEDPRICE FLOAT NOT NULL, L_DI
SCOUNT FLOAT NOT NULL, L_TAX FLOAT NOT NULL,
L_RETURNFLAG CHAR(1) NOT NULL, L_LI
NESTATUS 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)
PARTITIONING KEY (L_ORDERKEY) IN U
FTEMP1;

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

COMMIT WORK;
```

B.6 dss.ddl1TB.tbl.pok

```
CREATE TABLE TPCD.NATION  ( N_NATIONKEY  INTEGER NOT
NULL,
                    N_NAME       CHAR(25) NOT NULL,
                    N_REGIONKEY  INTEGER NOT NULL,
                    N_COMMENT    VARCHAR(152) NOT NULL with
default)
                    IN TS_NODE1;

CREATE TABLE TPCD.REGION  ( R_REGIONKEY  INTEGER NOT
NULL,
                    R_NAME       CHAR(25) NOT NULL,
                    R_COMMENT    VARCHAR(152) NOT NULL with
default)
                    IN TS_NODE1;

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

CREATE TABLE TPCD.SUPPLIER ( S_SUPPKEY     INTEGER NOT
NULL,
                    S_NAME       CHAR(25) NOT NULL,
                    S_ADDRESS    VARCHAR(40) NOT NULL,
                    S_NATIONKEY  INTEGER NOT NULL,
                    S_PHONE      CHAR(15) NOT NULL,
```

```
                S_ACCTBAL     FLOAT NOT NULL,
                S_COMMENT     VARCHAR(101) NOT NULL with
default)
    IN TPCDALL INDEX IN TPCDALL
          PARTITIONING KEY(S_SUPPKEY) USING HASHING;

CREATE TABLE TPCD.PARTSUPP ( PS_PARTKEY     INTEGER NOT
NULL,
                PS_SUPPKEY     INTEGER NOT NULL,
                PS_AVAILQTY    INTEGER NOT NULL,
                PS_SUPPLYCOST  FLOAT   NOT NULL,
                PS_COMMENT     VARCHAR(199) NOT NULL with
default)

    IN TPCDALL INDEX IN TPCDALL
          PARTITIONING KEY(PS_PARTKEY) USING HASHING ;

CREATE TABLE TPCD.CUSTOMER ( C_CUSTKEY     INTEGER NOT
NULL,
                C_NAME        CHAR(25) NOT NULL,
                C_ADDRESS     VARCHAR(40) NOT NULL,
                C_NATIONKEY   INTEGER NOT NULL,
                C_PHONE       CHAR(15) NOT NULL,
                C_ACCTBAL     FLOAT    NOT NULL,
                C_MKTSEGMENT  CHAR(10) NOT NULL,
                C_COMMENT     VARCHAR(117) NOT NULL with
default)
    IN TPCDALL INDEX IN TPCDALL
          PARTITIONING KEY(C_CUSTKEY) USING HASHING;

CREATE TABLE TPCD.ORDERS ( O_ORDERKEY       BIGINT NOT
NULL,
                O_CUSTKEY       INTEGER NOT NULL,
                O_ORDERSTATUS    CHAR(1) NOT NULL,
                O_TOTALPRICE     FLOAT NOT NULL,
                O_ORDERDATE      DATE NOT NULL,
                O_ORDERPRIORITY  CHAR(15) NOT NULL,
                O_CLERK          CHAR(15) NOT NULL,
                O_SHIPPRIORITY   INTEGER NOT NULL,
                O_COMMENT        VARCHAR(79) NOT NULL with
default)
    IN TPCDALL INDEX IN TPCDALL
          PARTITIONING KEY(O_ORDERKEY) USING HASHING;

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

COMMIT WORK;
```

B.7 dss.ldconfig1TB.pok.dbm

```
update dbm cfg using INTRA_PARALLEL on MAX_QUERYDEGREE
any CPUSPEED -1 SHEAPTHRE
S 100000 NUM_POOLAGENTS 0 DISCOVER_COMM TCPIP
COMM_BANDWIDTH 5 ;
```

B.8 dss.ldconfig1TB.pok

```
update db cfg for tpcd1tb using DFT_DEGREE 4 DFT_QUERYOPT 7
DBHEAP 2048 BUFFPAGE
 230000 APP_CTL_HEAP_SZ 64 SORTHEAP 15000 PCKCACHESZ
128 LOCKLIST 640 MAXLOCKS 4
0 NUM_IOCLEANERS 4 NUM_IOSERVERS 35 DFT_PREFETCH_SZ
128 MAXAPPLS 800 MAXFILOP 10
00;
```

B.9 doload.ksh

```
#! /bin/ksh
#    loadeverything
echo "load time summary:  " > /u/tpcd/tmp/loadstatus

echo "loading supplier at "‘date‘ >> /u/tpcd/tmp/loadstatus
dsh -a "/u/tpcd/tpcd/1TBstuff/loadfile supplier "
#sleep 10
echo "loading customer at "‘date‘ >> /u/tpcd/tmp/loadstatus
dsh -a "/u/tpcd/tpcd/1TBstuff/loadfile customer pipe "
#sleep 10
echo "loading part at "‘date‘ >> /u/tpcd/tmp/loadstatus
dsh -a "/u/tpcd/tpcd/1TBstuff/loadfile part pipe "
#sleep 10
echo "loading partsupp at "‘date‘ >> /u/tpcd/tmp/loadstatus
dsh -a "/u/tpcd/tpcd/1TBstuff/loadfile partsupp "
#sleep 10
echo "loading orders at "‘date‘ >> /u/tpcd/tmp/loadstatus
dsh -a ". /u/tpcd/tpcd/1TBstuff/loadfile orders pipe "
#sleep 10
echo "loading lineitem at "‘date‘ >> /u/tpcd/tmp/loadstatus
dsh -a ". /u/tpcd/tpcd/1TBstuff/loadfile lineitem "
#sleep 10
echo "loading nation and region at "‘date‘ >> /u/tpcd/tmp/loadstatus
dsh -w db2fr1n02 "/u/tpcd/tpcd/1TBstuff/loadtiny "
echo "finished loading at "‘date‘ >> /u/tpcd/tmp/loadstatus

cat /u/tpcd/tmp/loadstatus

#db2 connect to tpcd1tb;
#echo "sanity checking database";
#db2 -f sanity.sql
#db2 "select count(*) from tpcd.partsupp;"
#echo "sanity checking done"
#db2 connect reset;
#db2 terminate;
#echo "sleeping 15 seconds... load done"
#sleep 15
```

B.10 loadfile

```
#!/bin/ksh
TABLENAME=$1

FRSPACE=""
#if [[ "$TABLENAME" = "lineitem" ]]
#then
#FRSPACE=" totalfreespace=3 "
#echo $TABLENAME $FRSPACE
#fi
#if [[  "$TABLENAME" = "orders" ]]
#then
#FRSPACE=" totalfreespace=3 "
#echo $TABLENAME $FRSPACE
#fi

if [[ $# -lt 2 ]]
then
```

```
PIPE=""
else
PIPE=$2
fi
DATABASE=tpcd1tb
HOSTNAME=$(netstat -i|grep db2sn|awk '{print $4}')
NODE=$(grep $HOSTNAME /u/tpcd/sqllib/db2nodes.cfg|awk '{print
$1}')
. /u/tpcd/.profile
. /u/tpcd/sqllib/db2profile
export DB2NODE=$NODE
LOADFILE=/load/$TABLENAME$PIPE.0$NODE
echo Loading table $TABLENAME on node $NODE using $LOADFILE
if [ "$PIPE" != "" ]
then
echo creating named pipe $LOADFILE
rm  $LOADFILE
mknod $LOADFILE p
zcat /load/$TABLENAME.0$NODE.Z > $LOADFILE &
fi
db2 connect to $DATABASE
db2 "load from $LOADFILE of del modified by coldel| fastparse
usedefaults $FRSPA
CE dumpfile /tmp/tpcd/dmp$TABLENAME messages
/tmp/tpcd/msg$TABLENAME remote file
 /tmp/tpcd/rmt$TABLENAME replace into TPCD.$TABLENAME
nonrecoverable "
if [ "$PIPE" != "" ]
then
echo removing named pipe $LOADFILE
rm  $LOADFILE
fi
db2 connect reset
db2 terminate
```

B.11 loadtiny

```
#! /bin/ksh
# load nation and region on current node
. /u/tpcd/.profile
. /u/tpcd/sqllib/db2profile
HOSTNAME=$(netstat -i|grep db2sn|awk '{print $4}')
NODE=$(grep $HOSTNAME /u/tpcd/sqllib/db2nodes.cfg|awk '{print
$1}')
export DB2NODE=$NODE
echo Loading nation and region tables on node $NODE
db2 connect to tpcd1tb;
db2 "load from /load/nation.tbl of del modified by coldel| fastparse
noheader re
place into TPCD.nation nonrecoverable";
db2 commit work;
db2 "load from /load/region.tbl of del modified by coldel| fastparse
noheader re
place into TPCD.region nonrecoverable";
db2 commit work;
db2 connect reset;
db2 terminate;
```

B.12 dss.index.all.include

```
CREATE UNIQUE INDEX "TPCD  "."C_MS_CK" ON "TPCD
"."CUSTOMER"
            ("C_MKTSEGMENT" ASC,
             "C_CUSTKEY" ASC)
            PCTFREE 0 ;
commit work;

CREATE UNIQUE INDEX "TPCD    "."C_NAT_CKEY_REG" ON
"TPCD    "."CUSTOMER"
            ("C_NATIONKEY" ASC,
             "C_CUSTKEY" ASC)
            PCTFREE 0 ;
commit work;

create unique index tpcd.c_ck on tpcd.customer (c_custkey) pctfree 0;
commit;

CREATE INDEX "TPCD    "."L_OKCDRD" ON "TPCD    "."LINEITEM"
            ("L_ORDERKEY" ASC,
             "L_COMMITDATE" ASC,
             "L_RECEIPTDATE" ASC)
            PCTFREE 3 ;
commit work;

CREATE INDEX TPCD.L_SMRDCDSDOK ON TPCD.LINEITEM
            (L_SHIPMODE ASC,
             L_RECEIPTDATE ASC,
             L_COMMITDATE ASC,
             L_SHIPDATE ASC,
             L_ORDERKEY ASC)
            PCTFREE 4;
commit work;

CREATE INDEX "TPCD    "."L_SDDSLQEPSKPK" ON "TPCD
"."LINEITEM"
            ("L_SHIPDATE" ASC,
             "L_DISCOUNT" ASC,
             "L_QUANTITY" DESC,
             "L_EXTENDEDPRICE" ASC,
             "L_SUPPKEY" ASC,
             "L_PARTKEY" ASC)
            PCTFREE 5 ;
commit work;

CREATE INDEX "TPCD    "."PXL@OKSDRFSKEPDC" ON "TPCD
"."LINEITEM"
            ("L_ORDERKEY" ASC,
             "L_SHIPDATE" ASC,
             "L_RETURNFLAG" ASC,
             "L_SUPPKEY" ASC,
             "L_EXTENDEDPRICE" ASC,
             "L_DISCOUNT" ASC)
            PCTFREE 4 ;
commit work;

CREATE INDEX "TPCD    "."L_PKSKOKEPDSQN" ON "TPCD
"."LINEITEM"
            ("L_PARTKEY" ASC,
             "L_SUPPKEY" ASC,
             "L_ORDERKEY" ASC,
             "L_EXTENDEDPRICE" ASC,
             "L_DISCOUNT" ASC,
             "L_QUANTITY" ASC)
            PCTFREE 4 ;
commit work;

CREATE UNIQUE INDEX "TPCD    "."N_NAMNATKEY" ON "TPCD
"."NATION"
            ("N_NAME" ASC,
             "N_NATIONKEY" ASC)
            PCTFREE 0 ;
commit work;

CREATE UNIQUE INDEX "TPCD    "."N_NATKEYNAM" ON "TPCD
"."NATION"
            ("N_NATIONKEY" ASC) include(
             "N_NAME" ASC)
            PCTFREE 0 ;
commit work;

CREATE UNIQUE INDEX "TPCD    "."N_REGKEYNATKEYNAM" ON
"TPCD    "."NATION"
            ("N_REGIONKEY" ASC,
             "N_NATIONKEY" ASC) include(
```

```
            "N_NAME" ASC)
            PCTFREE 0 ;
commit work;

CREATE UNIQUE INDEX "TPCD  "."O_CK_OD_OK_SP" ON "TPCD
"."ORDERS"
            ("O_CUSTKEY" ASC,
             "O_ORDERDATE" ASC,
             "O_ORDERKEY" ASC) include(
             "O_SHIPPRIORITY" ASC)
            PCTFREE 3 ;
commit work;

CREATE UNIQUE INDEX "TPCD  "."O_OK_OD_OP" ON "TPCD
"."ORDERS"
            ("O_ORDERKEY" ASC) include(
             "O_ORDERDATE" ASC,
             "O_ORDERPRIORITY" ASC)
            PCTFREE 3 ;
commit work;

CREATE INDEX "TPCD    "."O_CLERK" ON "TPCD    "."ORDERS"
            ("O_CLERK" ASC)
            PCTFREE 3 ;
commit work;

CREATE UNIQUE INDEX "TPCD    "."O_OD_OK_OP_CK" ON "TPCD
"."ORDERS"
            ("O_ORDERDATE" ASC,
             "O_ORDERKEY" ASC) include(
             "O_ORDERPRIORITY" ASC,
             "O_CUSTKEY" ASC)
            PCTFREE 4 ;
commit work;

CREATE UNIQUE INDEX "TPCD  "."P_TP_PK" ON "TPCD    "."PART"
            ("P_TYPE" ASC,
             "P_PARTKEY" ASC)
            PCTFREE 0 ;
commit work;

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

CREATE unique INDEX TPCD.p_szpkbrty ON TPCD.part
            (p_size ASC,
            p_partkey ASC) include(
            p_brand  ASC,
            p_type ASC)
            PCTFREE 0;
commit work;

CREATE UNIQUE INDEX "TPCD    "."SXP_BRC2PK" ON "TPCD
"."PART"
            ("P_BRAND" ASC,
             "P_CONTAINER" ASC,
             "P_PARTKEY" ASC)
            PCTFREE 0 ;
commit work;

CREATE UNIQUE INDEX "TPCD    "."UXP_NMPK" ON "TPCD
"."PART"
            ("P_NAME" ASC,
             "P_PARTKEY" ASC)
            PCTFREE 0 ;
commit work;

CREATE UNIQUE INDEX "TPCD    "."UXP_SZTYPKMF" ON "TPCD
"."PART"
            ("P_SIZE" ASC,
             "P_TYPE" ASC,
             "P_PARTKEY" ASC) include(
             "P_MFGR" ASC)
            PCTFREE 0 ;
commit work;

CREATE UNIQUE INDEX "TPCD    "."UXPS_PK2KSC" ON "TPCD
"."PARTSUPP"
            ("PS_PARTKEY" ASC,
             "PS_SUPPKEY" ASC) include(
             "PS_SUPPLYCOST" ASC)
            PCTFREE 0 ;
commit work;

CREATE UNIQUE INDEX "TPCD    "."UXPS_SK2PKSCAQ" ON
"TPCD    "."PARTSUPP"
            ("PS_SUPPKEY" ASC,
             "PS_PARTKEY" ASC) include(
             "PS_SUPPLYCOST" ASC,
             "PS_AVAILQTY" ASC)
            PCTFREE 0 ;
commit work;

CREATE UNIQUE INDEX "TPCD    "."R_NAMREGKEY" ON "TPCD
"."REGION"
            ("R_NAME" ASC,
             "R_REGIONKEY" ASC)
            PCTFREE 0 ;
commit work;

CREATE UNIQUE INDEX "TPCD    "."PK_S_SUPPKEY" ON "TPCD
"."SUPPLIER"
            ("S_SUPPKEY" ASC)
            PCTFREE 0 ;
commit work;

CREATE UNIQUE INDEX "TPCD    "."S_NAT_SKEY_REG" ON
"TPCD    "."SUPPLIER"
            ("S_NATIONKEY" ASC,
             "S_SUPPKEY" ASC)
            PCTFREE 0 ;
commit work;

CREATE UNIQUE INDEX "TPCD    "."UXS_SKNK" ON "TPCD
"."SUPPLIER"
            ("S_SUPPKEY" ASC) include(
             "S_NATIONKEY" ASC)
            PCTFREE 0 ;
commit work;

select name,create_time from sysibm.sysindexes where tbcreator='TPCD';
commit;
```

B.13 dss.runstats.line

```
--CONNECT TO TPCD;

values current timestamp;

RUNSTATS ON TABLE TPCD.LINEITEM  AND DETAILED INDEXES
ALL;
values current timestamp;
COMMIT WORK;

--CONNECT RESET;
--TERMINATE;
```

B.14 dss.runstats.notline

```
CONNECT TO TPCD1TB;
```

```
values current timestamp;

RUNSTATS ON TABLE TPCD.NATION    AND DETAILED INDEXES
ALL;
commit;
values current timestamp;

RUNSTATS ON TABLE TPCD.REGION    AND DETAILED INDEXES
ALL;
commit;
values current timestamp;

RUNSTATS ON TABLE TPCD.SUPPLIER  AND DETAILED INDEXES
ALL;
commit;
values current timestamp;

RUNSTATS ON TABLE TPCD.PART      AND DETAILED INDEXES
ALL;
commit;
values current timestamp;

RUNSTATS ON TABLE TPCD.PARTSUPP  AND DETAILED
INDEXES ALL;
commit;
values current timestamp;

RUNSTATS ON TABLE TPCD.CUSTOMER  AND DETAILED
INDEXES ALL;
commit;
values current timestamp;

RUNSTATS ON TABLE TPCD.ORDERS    AND DETAILED INDEXES
ALL;
commit;
values current timestamp;

CONNECT RESET;
TERMINATE;
```

B.15 dss.ri3

```
values(current timestamp);
alter table TPCD.NATION add constraint pk primary key
(N_NATIONKEY);
commit work;

values(current timestamp);
alter table TPCD.CUSTOMER add constraint pk primary key
(C_CUSTKEY);
commit work;
values(current timestamp);

lock table tpcd.nation in exclusive mode;

with q(x) as
 (select 1 from tpcd.customer
  where c_nationkey not in
    (select n_nationkey from tpcd.nation))
select case x
        when 1 then raise_error('77777','Constraint Violated')
             else x end from q;
set constraints for tpcd.customer off;
values(current timestamp);
alter table TPCD.CUSTOMER add constraint fk foreign key
(C_NATIONKEY) references
 TPCD.NATION;
set constraints for tpcd.customer all immediate unchecked;
commit work;

values(current timestamp);
lock table tpcd.customer in exclusive mode;

with q(x) as
 (select 1 from tpcd.orders
  where o_custkey not in
    (select c_custkey from tpcd.customer))
select case x
        when 1 then raise_error('77777','Constraint Violated')
             else x end from q;
set constraints for tpcd.orders off;
values(current timestamp);
alter table TPCD.ORDERS add constraint fk foreign key (O_CUSTKEY)
references TPC
D.CUSTOMER ON DELETE CASCADE;
values(current timestamp);
set constraints for tpcd.orders all immediate unchecked;
commit work;
values(current timestamp);
```

B.16 dss.dbconfig1TB.pok.dbm

```
update dbm cfg using INTRA_PARALLEL on MAX_QUERYDEGREE
any SHEAPTHRES 90000 MAXAGENTS 2500 NUM_POOLAGENTS
50 NUM_INITAGENTS 50 DISCOVER_COMM TCPIP
COMM_BANDWIDTH 5 CPUSPEED 1.251712e-06 DIAGPATH
/tmp/db2dump FCM_NUM_RQB 15360 FCM_NUM_CONNEC
T 15000 DIAGLEVEL 3;
```

B.17 dss.dbconfig1TB.pok

```
update db cfg for tpcd1tb using DFT_DEGREE 4 DFT_QUERYOPT 7
DBHEAP 2048 BUFFPAGE
 275000 APP_CTL_HEAP_SZ 512 SORTHEAP 15000 PCKCACHESZ
1024 LOCKLIST 8192 MAXLOCK
S 13 NUM_IOCLEANERS 8 CHNGPGS_THRESH 60
NUM_IOSERVERS 24 DFT_PREFETCH_SZ 128 MAX
APPLS 2400 MAXFILOP 1000 UTIL_HEAP_SZ 8000 ;
```

Appendix C Queries and Updates

C.1 Qualification Queries and Output

```
*****************************
QUERY 1
*****************************
Start timestamp  05/01/98 13:54:36

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

Tag: Q1       Sequence number: 1

SELECT
L_RETURNFLAG,
L_LINESTATUS,
SUM(L_QUANTITY) AS SUM_QTY,
SUM(L_EXTENDEDPRICE) AS SUM_BASE_PRICE,
SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT)) 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 DAYS
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                3773034.000       5319329289.680       50
53976845.784       5256336547.676             25.510            35964.013
           0.050                   147907.
N            F                 100245.000        141459686.100        1
34380852.769        139710306.872               25.625            36160.451
           0.050                     3912.
N            O                7464940.000       10518546073.980       99
92072944.461       10392414192.063             25.542            35990.126
           0.050                   292262.
R            F                3779140.000       5328886172.990       50
62370635.934       5265431221.821             25.549            36025.461
           0.050                   147920.


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


Stop timestamp  05/01/98 13:54:41
*****************************
QUERY 2
*****************************
Start timestamp  05/01/98 13:54:45

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

Tag: Q2       Sequence number: 7

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_PA
RTKEY    P_MFGR              S_ADDRESS                         S_
PHONE        S_COMMENT

-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-----------------------------------
        9828.210  Supplier#000000647       UNITED KINGDOM
  13120  Manufacturer#5           jB16PyPyB7Bl52jMjSPw3mS
33
-258-202-4782  z1QhSiMj11Bm7COlLwh6Ql0B1R2Mg4CLn
LhiP0wiMzy72hlkP7l5in2y6RS6N13
0lz51nSRL5gOg5S26hPCCQN2L
        9508.370  Supplier#000000070       FRANCE
   3563  Manufacturer#1           M5C616R5h5SlMR3zzmLkSw24j2
16
-821-608-1166  m7z0CPShmBkhlChBAi3LkQ2CLw
mhl6QP362RPS3044CB2y41yhOhjlBin0CL7yhx
mhS4hBM07kQ1yyjOjz3C
        9508.370  Supplier#000000070       FRANCE
  17268  Manufacturer#4           M5C616R5h5SlMR3zzmLkSw24j2
16
-821-608-1166  m7z0CPShmBkhlChBAi3LkQ2CLw
mhl6QP362RPS3044CB2y41yhOhjlBin0CL7yhx
mhS4hBM07kQ1yyjOjz3C
        9453.010  Supplier#000000802       ROMANIA
  10021  Manufacturer#5
5yARQNSLNRAlOlBnkNQCik3SOlyClk7nmRhA2h0   29
-342-882-6463  65y3RQ2i0OP6Nz7mS hC
PxwLy7L1jQy6Ol63xO3iBCz52Rm1zm0MziCMLij2n6wk
y51mBOwx Qh52iz QB1545Amxyj
        9453.010  Supplier#000000802       ROMANIA
  13275  Manufacturer#4
5yARQNSLNRAlOlBnkNQCik3SOlyClk7nmRhA2h0   29
-342-882-6463  65y3RQ2i0OP6Nz7mS hC
PxwLy7L1jQy6Ol63xO3iBCz52Rm1zm0MziCMLij2n6wk
y51mBOwx Qh52iz QB1545Amxyj
        9192.100  Supplier#000000115       UNITED KINGDOM
  13325  Manufacturer#1
h0m3lzlSPMw2B0ny7LNyNCkjRRn7iyMlLBLA      33
```

-597-248-1220 1QzQjhSyx
ixm2lgz2Ry7075RL3MS5z36x56hxmR0wLN0LBxm164LzCMmALzOAj
n4
kz7i4wjOlCONl1C51M7nCMx66SBRAQA
9032.150 Supplier#000000959 GERMANY
4958 Manufacturer#4 205LNCzxMCnQ5gnz4n S3ynP6Mhnw
17
-108-642-3106 Px z7kOx56l7jQz NwBBQhky yM7kLgxRQw5zw6
426Bm55lC6 OkQ7hQPLixjM7y
47BNP16CRi0kjk354lgxh
8702.020 Supplier#000000333 RUSSIA
11810 Manufacturer#3
5iwkgN5n2BN15OmQk2602h0N6NzxPyiPN5lnj 32
-508-202-6136 SgimAjmn3wL7Rlxmh3LCwOPnhjyl 7xxzxAN
4ACx43y65NwQ7P

8615.500 Supplier#000000812 FRANCE
10551 Manufacturer#2 h4i2M2O0
ky1g2mlBOmxjzj0hA2h6nkSNhP 16
-585-724-6633 57i0NAyR0RP2jOh54C6B22OlSL

8615.500 Supplier#000000812 FRANCE
13811 Manufacturer#4 h4i2M2O0
ky1g2mlBOmxjzj0hA2h6nkSNhP 16
-585-724-6633 57i0NAyR0RP2jOh54C6B22OlSL

8488.530 Supplier#000000367 RUSSIA
6854 Manufacturer#4 nkmQ2Qzgh0wA 3x Sn2S7N5gmSOj
xwC COSn6 32
-458-198-9557 35C2RROP C Nlgi2N
SxAj0hQkn7kP5z4wSxSwgMxj6k4MRmh0S2Qm7R3z4jB OO
QBMl
8430.520 Supplier#000000646 FRANCE
11384 Manufacturer#3 61SjP6S y B0 32l11 16
-601-220-5489 kiw4NSNBNxy5kywzwyx0PMM21xiMOhxR423Akkm
Q7CNwRzQS23Nzz22 mnm6P377
Q3Mj7n 56BLm6lxwllh kSmN
8271.390 Supplier#000000146 RUSSIA
4637 Manufacturer#5 wh yPSk6hNBlB4l33iQ0wS0
RhBhQ4zQ3lz 32
-792-619-3155 jjwgwljRO63 n7OM2MP0hg3L1mlwBMLmMlS4Cgyn
LA5PwC2P0AS6g3C5mkOjO72N
Pig731m
8096.980 Supplier#000000574 RUSSIA
323 Manufacturer#4 hCOj4Cgx43xx jgP4QkL7gLN65
32
-866-246-8752 OhxNj6SlB56315B3k5SCBzwQyLk76zlj4Ow2Q
BC2wACkxh3S0RCyx6nARzSQR201
0k0BCPhOg6yQm
7392.780 Supplier#000000170 UNITED KINGDOM
7655 Manufacturer#2 PCxjjzNQihLNxgLw0SiMmQ
33
-803-340-5398 M116S1xzg54iC3k7OPLQi3Cimhghz2BClQk
g5Ag12QSBlhglANnw4MR MBS 72A

7205.200 Supplier#000000477 GERMANY
10956 Manufacturer#5 Mimj6403h zmAzAgg Bjy05O 2z
17
-180-144-7991 yRlyR SnMxmhPjAmBw
S02AxQ6yOhBRlOwzmlxzO0A2Sx075kjlAknn7z2 O0S7hy
0BiknwOQm6Pmz3gL4gj2z7
6820.350 Supplier#000000007 UNITED KINGDOM
13217 Manufacturer#5 z45m2jBRzl5ilLNz4 33
-990-965-2201 1PhngjmiSQl0RzRACP0l4S70xSL
QPSBM16072SkMLCgm4OOMjARLNQk3g1P3BB32
AgBMl462B0CP7Rh24
6721.700 Supplier#000000954 FRANCE
4191 Manufacturer#3 OM7xnNxNnkgQ mzh2g3RQmg1g
16
-537-341-8517 5ni3yCkmz5ymx0kCg74zhLA B516Si1w152AkiByx1Nl
NgghAkkmNz1jASj4mxzx
znOySg7hAyM3MRRnBj

6329.900 Supplier#000000996 GERMANY
10735 Manufacturer#2
k6135gA3zPwNl7L3Rl45mlnACjngOQQBB300iyA 17
-447-811-3282 PBO7wjlQMm1h3AAA 1NQAl0kkijnkRNgQ0
mh1z6QS0gC5lP1 ykmzNR20OlN506A
RS0z3j
6173.870 Supplier#000000408 RUSSIA
18139 Manufacturer#1 Cni6 zR5C4lh104POx5h05
mg53CQ2Sw4SAM2M2x 32
-858-724-2950 10SxMOwhjON3khzQ124gNnyw7B4nL7ml4L5llSR

5364.990 Supplier#000000785 RUSSIA
13784 Manufacturer#4 71OnPzQkC2P1hRNRgjyQP4n1
32
-297-653-2203 kiiPQ3ik7R ykAhRx43Rw70L1Ok
7AMi3AjRw7lklwxwyiL6S2O1COyS4QB46m5M1
67mjMwCm0w
5069.270 Supplier#000000328 GERMANY
16327 Manufacturer#1 5O4033xSgml 17
-231-513-5721 OMk3ALAPNmj6lBLMAS7M1nCAS
4xLj51iy2klix3nPi26gAxPgANmk6zSi6 3A7m
l11BOwiC6xLB4hBRiPM
4941.880 Supplier#000000321 ROMANIA
7320 Manufacturer#5 hyLQ mg42S2kAMlj M3BwMSjS
29
-573-279-1406 y2644kMhOkPCm5P5y7Lmz7OR6mgSmBN631RggmC

4672.250 Supplier#000000239 RUSSIA
12238 Manufacturer#1 y4ymj7B5BN1nMSkwPPggAl
32
-396-654-6826 Py3RA2gykmSCmj0z3ii7Rxzhz6OyR RxS C3S23LPQ

4586.490 Supplier#000000680 RUSSIA
5679 Manufacturer#3 BP1Nlw5nPMxRnOAwm
32
-522-382-1620 kA0y25RNO1Al im7SyiPzSym3M5OS52l6S576kn0S2k
0mPBLlAzL6Ax7CM6iNi4C
gCy6Bln7hlhxmlRng
4518.310 Supplier#000000149 FRANCE
18344 Manufacturer#5 4B QSy5Bl2 16
-660-553-2456 hijkPhgL1g4L1Q27y0Q42wh0Qz3jPiL4NgkM4NNg1
llQ1yNNBk1C1QnlRO7 4ki

4315.150 Supplier#000000509 FRANCE
18972 Manufacturer#2 B5 iPRn7L4yMllgwCnRPMA
16
-298-154-3365 ygiPh7ymP7jBznmR2lQLLgjmi1wik

3526.530 Supplier#000000553 FRANCE
8036 Manufacturer#4 ylLOx2gMw 5iBl6AiNL60Q
16
-599-552-3755 L3ggShlRlyxmR4MNl7Rw7OQign6yO

3526.530 Supplier#000000553 FRANCE
17018 Manufacturer#3 ylLOx2gMw 5iBl6AiNL60Q
16
-599-552-3755 L3ggShlRlyxmR4MNl7Rw7OQign6yO

3294.680 Supplier#000000350 GERMANY
4841 Manufacturer#4 x5kRL2z1BPg0 BO 2hi1iOyh
30RRg0OPj 17
-113-181-4017 BjQznni44OmQ7S16yl3zxk2M6nM4M 27yMPML

2972.260 Supplier#000000016 RUSSIA
1015 Manufacturer#4 B7wLkSLRjNS MS1C 32
-822-502-4215 C7w6S6QzhAPQmMmNmMN1hA0llQOA
00m1NmC25wyQ461SA jy03zmRh22MLM00zhm
i
2963.090 Supplier#000000840 ROMANIA
3080 Manufacturer#2 1ynwiQkNh0
CMRRCk4l306M2ij0jykg6QNgSCAzy 29

-781-337-5584 S7NRMx43RmOjxML6hxLyN75LzxBwB0wjSLx3
S3Cwh52S6ilSOLhQm0 6C1 yzx3j
Pm6Sjg 5By0BCPwOR32i1CQgxR0gB43gh
2221.250 Supplier#000000771 ROMANIA
13981 Manufacturer#2 LAjCRjl3nAMzzhmw0Sx1Mg
29
-986-304-9006 jhk0N7NlhS23iCngC52BBC
0jilM0wByx0LB5R070R2lCx113lQiS7xNhBRA0xknl
NxLiA
1381.970 Supplier#000000104 FRANCE
18103 Manufacturer#3 i Qnl4 1 jiwM C2yxAylL5R4SBQh54N6
16
-434-972-6922 MwnBw1g71Pig2Am7nz0Mm5SNl7OwQLAkN56ji

906.070 Supplier#000000138 ROMANIA
8363 Manufacturer#4 1iBxSxLl1Mh3
6LS6PlLPNlnlMjCQh22z6n5 29
-533-434-6776 nLjQAmCw77R2jRMgz5LSyyx1QN l
4jMMO3RAkxOkzRmwQl3Qm5236k72RRPnim0
BkzQnBMM6A PMml2n
765.690 Supplier#000000799 RUSSIA
11276 Manufacturer#2 Am7yihz47mg NkgQL w By4
32
-579-339-1495 MMRPNQ
4l66mQQPNniAiiL0PQ2C4yyBRn1nRlxnxkj5Ak45Pw
mQk1ROhz66BRQii
L gPRQRy 56MyQ nS1N14R 7Ml6xhl2lOS3
727.890 Supplier#000000470 ROMANIA
6213 Manufacturer#3 gAySBM2N7 PgwP5kiP4n7BzOik0M
29
-165-289-1523 zCkPgn
6wN5A3R47gljlQ3hNSLShP2RALxCiinkOy4wCwA1LCiBO5yiSC
yBAA li
i
683.070 Supplier#000000651 RUSSIA
4888 Manufacturer#4
ymQ6PByCh4lzxBBPLB2wwOhRh47wQMQSPL 32
-181-426-4490
kx6jhQkwzg6RkRgPLPM30BgL1R726l1m5AMk0MmMQBQ
nCihiLxhMgCgRih6MmMx0
PglRQ7AQnl72g50
167.560 Supplier#000000290 FRANCE
2037 Manufacturer#1 xm6n30QmBn75QPh7Py01lxlnB4n1lNh
MBi1 16
-675-286-5102 gi6y41Bg5
AkhmCh30Qyxx515xLi6MRjCQ4n2xjML7N0PNSCyBPwS1C

91.390 Supplier#000000949 UNITED KINGDOM
9430 Manufacturer#2 zCmkwm43LQ62Q3O 4nkNA
33
-332-697-2768 Nm5yn23SxgPAMRz6B5CjAj04nkNx5lA7O

-314.060 Supplier#000000510 ROMANIA
17242 Manufacturer#4 hwx1kP6nQ0NgPO232nxAwL4QPjRN
i64jOiPjC0 29
-207-852-3454
5NAyRmwSCQ5hMB6RiN0Qyhjghnl0BPjPlNlP725AAOCw

-820.890 Supplier#000000409 GERMANY
2156 Manufacturer#5 Nk4z1lg25gNwMwO
2BnMOn1P3k0yA7i2l 17
-719-517-9836 CmLnngQywQ4hhx3266yz5Qgj
nOm1P1RC3gxmg4kmnNCSnM4PBO4hNQA1xyzC2LCQ
BP07w2z3
-845.440 Supplier#000000704 ROMANIA
9926 Manufacturer#5 zi jjmO4A57 mQgLmhALCM3B zL
29
-300-896-5991
OSh0PRm6ByyS2wLBNPQPPi2CL0zN0Px1LNOlwl6y0Rgij zP5hPCz
745nn62hPOR
l6i7SynBCC4Qy4Qkn1 l
-942.730 Supplier#000000563 GERMANY
5797 Manufacturer#1 xSLLy 4jm17OC 17
-108-537-2691
1gSOl3A0jnkAmQLllmygzmg7hhjjAmRxN5SwjxM16Ok61 wm5N42

Number of rows retrieved is: 44

Stop timestamp 05/01/98 13:54:45
****
QUERY 3
****
Start timestamp 05/01/98 13:54:51

Tag: Q3 Sequence number: 13

```
SELECT
CHAR(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
```

```
1 REVENUE O_ORDERDATE O_SHIPPRIORITY
-------------------------------------------------------------------------
260930 320547.253 1995-03-12 0
402497 298879.532 1995-02-12 0
457859 296490.675 1995-01-17 0
509889 294068.874 1995-02-03 0
58117 292632.833 1995-02-21 0
538311 279665.996 1995-03-07 0
588421 275477.117 1995-03-03 0
416167 273765.453 1995-02-22 0
97830 273227.061 1995-03-04 0
90276 272233.917 1995-03-04 0
```

Number of rows retrieved is: 10

Stop timestamp 05/01/98 13:54:52
****
QUERY 4
****
Start timestamp 05/01/98 13:54:41

Tag: Q4 Sequence number: 2

```
SELECT
O_ORDERPRIORITY,
COUNT_BIG(*) AS ORDER_COUNT
FROM TPCD.ORDERS
WHERE O_ORDERDATE >= DATE('1993-07-01')
AND O_ORDERDATE < DATE('1993-07-01') + 3 MONTHS
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                  999.
2-HIGH                   1002.
3-MEDIUM                  1021.
4-NOT SPECIFIED            997.
5-LOW                    1089.


Number of rows retrieved is:  5
-----------------------------------------------


Stop timestamp  05/01/98 13:54:42
*****************************
QUERY 5
*****************************
Start timestamp  05/01/98 13:54:52

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

Tag: Q5      Sequence number: 14

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
-------------------------------------------------
CHINA                  7349391.471
INDONESIA                 6485853.403
INDIA                 5505346.820
JAPAN                  5388883.594
VIETNAM                   4728846.602


Number of rows retrieved is:  5
-----------------------------------------------


Stop timestamp  05/01/98 13:54:56
*****************************
QUERY 6
*****************************
Start timestamp  05/01/98 13:54:44

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

Tag: Q6      Sequence number: 6

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
---------------------
      11450588.043


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


Stop timestamp  05/01/98 13:54:45
*****************************
QUERY 7
*****************************
Start timestamp  05/01/98 13:54:56

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

Tag: Q7      Sequence number: 16

SELECT
SUPP_NATION,
CUST_NATION,
YEAR,
SUM(VOLUME) AS REVENUE
FROM
(SELECT
N1.N_NAME AS SUPP_NATION,
N2.N_NAME AS CUST_NATION,
YEAR(L_SHIPDATE) AS YEAR,
L_EXTENDEDPRICE*(1-L_DISCOUNT) AS VOLUME
FROM TPCD.SUPPLIER, TPCD.LINEITEM, TPCD.ORDERS,
TPCD.CUSTOMER, TPCD.NATION N1, T
PCD.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, YEAR
ORDER BY SUPP_NATION, CUST_NATION, YEAR

SUPP_NATION           CUST_NATION           YEAR
REVENUE

-------------------------------------------------------------------------------
---------
FRANCE                GERMANY               1995        4611
421.440
FRANCE                GERMANY               1996        4828
420.372
GERMANY               FRANCE                1995        6755
766.841
GERMANY               FRANCE                1996        5810
951.396


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


Stop timestamp  05/01/98 13:54:59
*****************************
QUERY 8
```

```
****************************
Start timestamp  05/01/98 13:54:46

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

Tag: Q8       Sequence number: 10

SELECT
YEAR AS YEAR,
SUM(CASE WHEN NATION = 'BRAZIL'
THEN VOLUME
ELSE 0
END) / SUM(VOLUME) AS MKT_SHARE
FROM
(SELECT
YEAR(O_ORDERDATE) AS 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.N
ATION 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 YEAR
ORDER BY YEAR

YEAR        MKT_SHARE
-----------------------------------
     1995             0.055
     1996             0.089


Number of rows retrieved is:  2
------------------------------------------------



Stop timestamp  05/01/98 13:54:50
****************************
QUERY 9
****************************
Start timestamp  05/01/98 13:54:59

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

Tag: Q9       Sequence number: 17

SELECT
NATION,
YEAR,
SUM(AMOUNT) AS SUM_PROFIT
FROM
(SELECT
N_NAME AS NATION,
YEAR(O_ORDERDATE) AS YEAR,
L_EXTENDEDPRICE*(1-L_DISCOUNT)-
PS_SUPPLYCOST*L_QUANTITY AS AMOUNT
FROM TPCD.PART, TPCD.SUPPLIER, TPCD.LINEITEM,
TPCD.PARTSUPP, TPCD.ORDERS, TPCD.N
ATION
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, YEAR
ORDER BY NATION, YEAR DESC

NATION                YEAR        SUM_PROFIT
-------------------------------------------------------------
ALGERIA                 1998         1946316.005
ALGERIA                 1997         2973825.692
ALGERIA                 1996         3308881.517
ALGERIA                 1995         3092227.299
ALGERIA                 1994         3406958.710
ALGERIA                 1993         3140744.026
ALGERIA                 1992         3330704.407
ARGENTINA               1998         3045410.008
ARGENTINA               1997         4255378.593
ARGENTINA               1996         4651751.937
ARGENTINA               1995         4897797.003
ARGENTINA               1994         4823465.769
ARGENTINA               1993         4499810.713
ARGENTINA               1992         4764593.386
BRAZIL                  1998         2932051.363
BRAZIL                  1997         3784531.350
BRAZIL                  1996         3965665.690
BRAZIL                  1995         4063060.861
BRAZIL                  1994         4236277.350
BRAZIL                  1993         4363461.313
BRAZIL                  1992         4684749.233
CANADA                  1998         2217064.038
CANADA                  1997         2950110.610
CANADA                  1996         3184049.969
CANADA                  1995         3962540.195
CANADA                  1994         3365251.022
CANADA                  1993         3617013.367
CANADA                  1992         3407955.249
CHINA                   1998         3048192.023
CHINA                   1997         5001207.691
CHINA                   1996         4800958.313
CHINA                   1995         5154927.728
CHINA                   1994         5882634.534
CHINA                   1993         4733364.821
CHINA                   1992         5014704.079
EGYPT                   1998         1892538.744
EGYPT                   1997         3849220.075
EGYPT                   1996         3418656.553
EGYPT                   1995         3766170.603
EGYPT                   1994         3520025.559
EGYPT                   1993         4375424.745
EGYPT                   1992         4586034.394
ETHIOPIA                1998         1860117.728
ETHIOPIA                1997         3705722.334
ETHIOPIA                1996         3577215.393
ETHIOPIA                1995         3425219.552
ETHIOPIA                1994         3428616.185
ETHIOPIA                1993         3459815.431
ETHIOPIA                1992         3280072.908
FRANCE                  1998         1592531.548
FRANCE                  1997         2746176.538
FRANCE                  1996         2505844.880
FRANCE                  1995         2902077.004
FRANCE                  1994         2532229.560
FRANCE                  1993         2305725.442
FRANCE                  1992         2955126.689
GERMANY                 1998         3538625.734
GERMANY                 1997         4425943.399
GERMANY                 1996         4266344.955
GERMANY                 1995         3952963.516
GERMANY                 1994         4462655.798
GERMANY                 1993         4435094.657
GERMANY                 1992         4521715.412
INDIA                   1998         3378369.337
```

INDIA	1997	4186477.848
INDIA	1996	5074383.925
INDIA	1995	4487435.379
INDIA	1994	4718312.626
INDIA	1993	4499573.810
INDIA	1992	4712930.333
INDONESIA	1998	2902077.101
INDONESIA	1997	4973644.228
INDONESIA	1996	4977652.489
INDONESIA	1995	5359380.151
INDONESIA	1994	4854637.200
INDONESIA	1993	4213131.423
INDONESIA	1992	4999478.506
IRAN	1998	2415763.101
IRAN	1997	4227175.109
IRAN	1996	4527365.027
IRAN	1995	4139514.717
IRAN	1994	4166316.391
IRAN	1993	3366959.588
IRAN	1992	3599399.702
IRAQ	1998	2596922.633
IRAQ	1997	3707054.112
IRAQ	1996	3726138.383
IRAQ	1995	4350503.892
IRAQ	1994	4131512.791
IRAQ	1993	3787196.421
IRAQ	1992	4043738.134
JAPAN	1998	2265666.942
JAPAN	1997	3988819.281
JAPAN	1996	4319004.534
JAPAN	1995	4262698.637
JAPAN	1994	3545212.620
JAPAN	1993	4051565.975
JAPAN	1992	3692137.445
JORDAN	1998	1978591.742
JORDAN	1997	3315454.287
JORDAN	1996	3236531.980
JORDAN	1995	2778207.986
JORDAN	1994	2420301.072
JORDAN	1993	3272130.935
JORDAN	1992	2649126.086
KENYA	1998	2265677.727
KENYA	1997	3493019.323
KENYA	1996	3346373.296
KENYA	1995	3537360.325
KENYA	1994	2800950.716
KENYA	1993	3477468.302
KENYA	1992	2719618.041
MOROCCO	1998	2549499.930
MOROCCO	1997	3891824.898
MOROCCO	1996	3730777.735
MOROCCO	1995	3469641.134
MOROCCO	1994	3747593.208
MOROCCO	1993	3620742.698
MOROCCO	1992	4303609.249
MOZAMBIQUE	1998	2024719.461
MOZAMBIQUE	1997	3706003.087
MOZAMBIQUE	1996	3376430.930
MOZAMBIQUE	1995	2737631.643
MOZAMBIQUE	1994	3373146.481
MOZAMBIQUE	1993	3608300.374
MOZAMBIQUE	1992	3551263.950
PERU	1998	2142791.972
PERU	1997	4664076.154
PERU	1996	3623628.934
PERU	1995	3908939.791
PERU	1994	3386204.157
PERU	1993	3877048.489
PERU	1992	3768394.249
ROMANIA	1998	1760625.703
ROMANIA	1997	2707685.329
ROMANIA	1996	2553345.479
ROMANIA	1995	2715901.590
ROMANIA	1994	3023644.056
ROMANIA	1993	2873247.320
ROMANIA	1992	2728060.707
RUSSIA	1998	2975973.217
RUSSIA	1997	3785806.468
RUSSIA	1996	4217625.587
RUSSIA	1995	3883445.515
RUSSIA	1994	4395855.006
RUSSIA	1993	3900944.177
RUSSIA	1992	4691358.609
SAUDI ARABIA	1998	2931482.833
SAUDI ARABIA	1997	5498943.156
SAUDI ARABIA	1996	4473723.738
SAUDI ARABIA	1995	5939212.934
SAUDI ARABIA	1994	4527695.709
SAUDI ARABIA	1993	4928702.017
SAUDI ARABIA	1992	5527261.522
UNITED KINGDOM	1998	3198731.373
UNITED KINGDOM	1997	4363882.744
UNITED KINGDOM	1996	4730956.674
UNITED KINGDOM	1995	4842014.546
UNITED KINGDOM	1994	4912706.557
UNITED KINGDOM	1993	4415255.963
UNITED KINGDOM	1992	4375524.230
UNITED STATES	1998	1892045.160
UNITED STATES	1997	3102027.859
UNITED STATES	1996	3334320.258
UNITED STATES	1995	3168244.604
UNITED STATES	1994	3296960.101
UNITED STATES	1993	3558109.055
UNITED STATES	1992	2755129.388
VIETNAM	1998	2906627.025
VIETNAM	1997	4544560.448
VIETNAM	1996	4314258.999
VIETNAM	1995	4365340.861
VIETNAM	1994	3686987.712
VIETNAM	1993	3764237.179
VIETNAM	1992	3420922.004

Number of rows retrieved is: 175

Stop timestamp 05/01/98 13:55:02

****

QUERY 10

****

Start timestamp 05/01/98 13:54:42

Tag: Q10 Sequence number: 4

```
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 MONTHS
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

-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------------------------------------
-------------------------------------------------
      9722  Customer#000009722              464618.258            474.0
40  CANADA                l Mwzn4NAk6j                       13-518-
602-8070  5L 50Oy
RSgBAzPxmOSi5wk6xxOR7kh2nnPlgy7LBng2hOw5B01
RmCM120L24Pkg7PS1z
wC11BCnz4L6i15PkixP26l66
     12800  Customer#000012800              444265.642            1900.8
40  PERU                  57zjB3CQx4P4OB2R2MBi2mwhSllM4mn 4 nC6
27-142-
205-3552  0hwglS77RB56Rx436lQ0N16CxhOPnmyhgwz
5z64wnj1kiC4jL350mM41y71hNxBllPjyA
4hiN1wzjjM7SCxAN244mk2A
      1025  Customer#000001025              442028.022            3363.4
60  INDIA                 lkiSn154M5ROi                      18-588-
456-4616  0Bl45z233Rniw0OO064nPBgP16kimO0y74iLh73g1N4 m310
jQ yQzPA50iC 3MA75g2B
j162Nw4P
     13028  Customer#000013028              441692.240            -452.6
60  UNITED KINGDOM        yP714ORSNgNN2LA3L5B
33-253-
660-2127  xPkmnhL2BkhkNyww4khlxwwAymN
h11PSjBCNMi50LkyOhO6CC 5nzOQCALzliOk2R66w
l05hRPO3iSP
      3694  Customer#000003694              438180.070            2960.4
40  UNITED KINGDOM        2CCklmCBOCC                  33-
421-
331-3127  MzLxQxLlLx3MPxlAwg1B5kg61zxkPnk
xiAm6PhMMAAQ2nzN3S6zzgP x70w0lhhPx4QRz
lMMy0204lAl3mBO7jh2jAP0N60wg367z
       976  Customer#000000976              435897.632            7772.8
50  ROMANIA               QzR 56Px1kgS  wANnAz02RS 30n Pm
29-436-
660-4732  kzn32776 gwzkMzzzO4yxOAnkR7hR4R4x2SMwilz3x6h
nN7OnNLRMml3 kz5SLwi1yklO
xiwS4g0wmA5A 4hmgBSwRRiQ1
     8206  Customer#000008206              429905.110            6046.3
60  ARGENTINA             P yMg30BBBBx NMgC03AmzN2
11-571-
859-1370  hLi122RMPmLC36Oy0kxO71zz2wCR0QQCl7z26hlQ3mM

     13532  Customer#000013532              427731.804            -924.1
80  KENYA
6ij7M5PBMx2kwwyz62Oj4SL5S0mRCwl3m1Rmw     24-525-
332-7244  7ih7yRz214zO67AiNPx64nO5l5k
yj6i3jLA5PCL15Q4QlA3ll60iM1P iBxCixg6 1hCh
2RCnjOzk5R OnO 1OhhC3m4631m5
     12745  Customer#000012745              422327.693            9691.3
30  CHINA                 SgS1LMC4gB2NM3wh                   28-985-
189-6174  jl72wjSw0 S6 7L4Cgxw PkyO5Nl2LL7LBR

      2344  Customer#000002344              411240.109            5597.2
20  MOROCCO               O3PC7ikBgw OAzPAlm2P
426zm3BnBN6Q1O 6N    25-593-
745-7663  5NBn0wRNngLw2z5kyn1AhL0ASyg6SMhM
i2kMOyxARAnlO0Q5j4CBNARix7ABlMAC

      2656  Customer#000002656              401185.952            8115.5
50  ALGERIA               On55lAS3Rm5RxS m                   10-667-
469-8092  46ABx4jgni mlBMPCLxRhyPQM4RNS 5yO1L7zSOmk
MhPxAxQQ6lQnLj 17LymOhi4l5in
nzOyB2Olxzmw3gmx0SxiyBN5CSMNgCkLCkMgO
       59  Customer#000000059              400759.150            3458.6
00  ARGENTINA             wP6CMyClly0lS4CAM1mzm
11-355-
584-3112  l1g7xBCxxC7SM
5AkmnmAk00677O1MzA2R7A0Cx0Njixj56jL2iN
PNkSNQiy55m6ki3Og
nhM47mSR7B
      7069  Customer#000007069              396217.520            8198.9
40  INDONESIA             55Cw7ChL4Bi5ONn2A4m2i2n4nSNQQMjml
19-644-
744-1798  6jNS624l75zlxNli4lxO5zyPykPS1xnilS0NhkgOAkSx7P

      6553  Customer#000006553              385863.595            8985.9
00  MOZAMBIQUE            R3LnnxONBjCLC0MRkxy7
26-166-
724-4677  S7CkNLwA3kh006j71lwAlC25Bw6AMQ6i
6C0OSS6O7ARNNny60Ogh  3642mRxyiAgy5yk
 3nPO4473wkNg5R6gzO4lz3zmM2m7MiLAilCC
      3095  Customer#000003095              384246.108            8829.2
10  IRAQ                  S1gMCnBLwzi mCgB664
jl00Ll1Snh1iPMgCgR5   21-847-
218-8188  3LSx7PxS
A4A5Cl3gAy3mg4Qj2xQlyx7xM1kA664AM7zmMmzORh3C1h
MO3nw6MymiljAM
g65hOMB4Sn44kO w0lin7
      3391  Customer#000003391              382541.776            7742.3
50  CANADA                m3 CORmQNLzkShymLS iMkCimRSl20 NB
13-592-
494-2668  ynMlmhMBA5ikC1nCghlmAhQ0 675S3y2R33yjkNPQOS

     13678  Customer#000013678              376280.556            9030.4
00  MOROCCO               BMk77lQm1lwNA0LghAkg3hCwNl4
25-306-
951-3937
mOS55RASx1wP136nQ5xBLznLhgw1kQ6PO6imNxQ7kR0x71P0SzBy
Mzh

      6062  Customer#000006062              374512.654            1370.3
50  CANADA                n5zzil60zyxAlkzx7x1nihigPzR OBkR
znMOMh   13-756-
700-4918  4zAm4wNB
li4QRPgPz2wM541x043hmLj4O3LBkALCP16hj2RQBO1OMNly7ww1Q
P7w5i SS
n0jNhAR yQmmz1hi5j3
       554  Customer#000000554              373004.470            8395.5
70  BRAZIL                jC5zhQky4zQB27lB5Sm AQhQ  Px0          12-
938-
503-7317  0nxCl3 xSmiLQO 1M
2n0NCiRlnMMxP25j26x2igLhNOxjgMgmwwy7OkjzCACOg0z2LAjO
m0RPRmOPiCAAQwLlQSg 1yS3 gLCM1M2BzjnSjPl3nwAkk
     13126  Customer#000013126              371722.001            6172.9
10  INDIA                 xPAS4MnPh40i5Q2h4NQ61zz4RkyAwANA
18-288-
190-4145  nniMkAN6C0ClQ0mMmPz27liz4hk6L
2MlwPxh42N1l0R2hRwxxzlwMkxO4MAyz7RCj43Nx
LwQ3m6P27yAj


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


Stop timestamp  05/01/98 13:54:44
****************************
QUERY 11
****************************
Start timestamp  05/01/98 13:54:44

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

Tag: Q11 Sequence number: 5

```
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) * 0.0010000000
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
12098	16227681.210
5134	15709338.520
13334	15023662.410
17052	14351644.200
3452	14070870.140
12552	13332469.180
1084	13170428.290
5797	13038622.720
12633	12892561.610
403	12856217.340
1833	12024581.720
2084	11502875.360
17349	11354213.050
18427	11282385.240
2860	11262529.950
17852	10934711.930
9871	10889253.680
12231	10841131.390
6366	10759786.810
12146	10257362.660
5043	10226395.880
12969	10125777.930

Number of rows retrieved is: 22

Stop timestamp 05/01/98 13:54:44

****

QUERY 12

****

Start timestamp 05/01/98 13:54:50

Tag: Q12 Sequence number: 11

```
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	654	950
SHIP	684	1004

Number of rows retrieved is: 2

Stop timestamp 05/01/98 13:54:50

****

QUERY 13

****

Start timestamp 05/01/98 13:54:56

Tag: Q13 Sequence number: 15

```
SELECT YEAR,
SUM(REVENUE) AS REVENUE
FROM
(SELECT YEAR(O_ORDERDATE) AS YEAR,
L_EXTENDEDPRICE*(1-L_DISCOUNT) AS REVENUE
FROM TPCD.LINEITEM, TPCD.ORDERS
WHERE O_ORDERKEY = L_ORDERKEY
AND O_CLERK = 'Clerk#000000088'
AND L_RETURNFLAG = 'R') AS PERFORMANCE
GROUP BY YEAR
ORDER BY YEAR
```

YEAR	REVENUE
1992	1262855.731
1993	964121.033
1994	1750395.294
1995	198820.299

Number of rows retrieved is: 4

Stop timestamp 05/01/98 13:54:56

****

QUERY 14

****

Start timestamp 05/01/98 13:54:46

Tag: Q14 Sequence number: 9

```
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
----------------------
            16.729


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


Stop timestamp  05/01/98 13:54:46
*****************************
QUERY 15
*****************************
Start timestamp  05/01/98 13:54:42

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

Tag: Q15c     Sequence number: 3

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 MONTHS
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
-------------------------------------------------------------------------------
-----------------------------------------
       389  Supplier#000000389        PB1Lx0xx6LMz3h7Rx63m6j3QmMx
  34-885-883-5717          1418538.214


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


Stop timestamp  05/01/98 13:54:42
*****************************
QUERY 16
*****************************
Start timestamp  05/01/98 13:54:45

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

Tag: Q16      Sequence number: 8

SELECT
P_BRAND,
P_TYPE,
P_SIZE,
COUNT_BIG(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 '%Better Business Bureau%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#14   SMALL ANODIZED NICKEL            45
12.
Brand#22   SMALL BURNISHED BRASS            19
12.
Brand#25   PROMO POLISHED COPPER            14
12.
Brand#35   LARGE ANODIZED STEEL             45
12.
Brand#35   PROMO BRUSHED COPPER              9
12.
Brand#51   ECONOMY ANODIZED STEEL            9
12.
Brand#53   LARGE BRUSHED NICKEL             45
12.
Brand#11   ECONOMY POLISHED COPPER          14
 8.
Brand#11   LARGE PLATED STEEL              23
 8.
Brand#11   PROMO POLISHED STEEL            23
 8.
Brand#11   STANDARD ANODIZED COPPER          9
 8.
Brand#12   ECONOMY BURNISHED BRASS          9
 8.
Brand#12   LARGE ANODIZED BRASS            14
 8.
Brand#12   SMALL ANODIZED TIN              23
 8.
Brand#12   SMALL BRUSHED NICKEL            23
 8.
Brand#12   STANDARD ANODIZED BRASS          3
 8.
Brand#12   STANDARD BURNISHED TIN          23
 8.
Brand#13   ECONOMY POLISHED BRASS           9
 8.
Brand#13   LARGE BURNISHED COPPER           45
 8.
Brand#13   MEDIUM ANODIZED STEEL           23
 8.
Brand#13   MEDIUM PLATED NICKEL             3
 8.
Brand#13   PROMO BURNISHED BRASS            9
 8.
Brand#13   PROMO POLISHED BRASS             3
 8.
Brand#13   PROMO POLISHED TIN              36
 8.
Brand#13   SMALL BURNISHED STEEL           23
 8.
Brand#13   STANDARD BRUSHED STEEL           9
 8.
Brand#14   ECONOMY BRUSHED TIN              3
 8.
Brand#14   ECONOMY BURNISHED TIN           23
 8.
Brand#14   PROMO BRUSHED STEEL              9
 8.
```

Brand#14	PROMO PLATED TIN	45
Brand#15	ECONOMY PLATED TIN	9
Brand#15	STANDARD BRUSHED COPPER	14
Brand#15	STANDARD PLATED TIN	3
Brand#21	ECONOMY POLISHED TIN	3
Brand#21	PROMO POLISHED COPPER	9
Brand#21	PROMO POLISHED TIN	49
Brand#21	STANDARD PLATED BRASS	49
Brand#21	STANDARD PLATED NICKEL	49
Brand#22	ECONOMY ANODIZED TIN	49
Brand#22	ECONOMY BRUSHED BRASS	14
Brand#22	LARGE BURNISHED TIN	36
Brand#22	MEDIUM ANODIZED STEEL	36
Brand#22	MEDIUM PLATED STEEL	9
Brand#22	PROMO POLISHED NICKEL	9
Brand#22	SMALL ANODIZED STEEL	19
Brand#22	STANDARD ANODIZED COPPER	23
Brand#23	ECONOMY BRUSHED NICKEL	23
Brand#23	LARGE ANODIZED BRASS	9
Brand#23	LARGE ANODIZED STEEL	23
Brand#23	SMALL BRUSHED COPPER	23
Brand#23	STANDARD BRUSHED TIN	3
Brand#23	STANDARD BURNISHED NICKEL	49
Brand#23	STANDARD PLATED NICKEL	36
Brand#24	ECONOMY ANODIZED BRASS	19
Brand#24	ECONOMY POLISHED BRASS	36
Brand#24	LARGE BURNISHED STEEL	14
Brand#24	MEDIUM PLATED NICKEL	36
Brand#25	ECONOMY BRUSHED STEEL	49
Brand#25	MEDIUM BURNISHED TIN	3
Brand#25	PROMO ANODIZED TIN	36
Brand#25	PROMO PLATED NICKEL	3
Brand#25	SMALL BURNISHED BRASS	3
Brand#31	LARGE ANODIZED BRASS	3
Brand#31	SMALL ANODIZED COPPER	3
Brand#31	SMALL ANODIZED NICKEL	9
Brand#31	SMALL ANODIZED STEEL	14
Brand#32	MEDIUM ANODIZED STEEL	49
Brand#32	MEDIUM BURNISHED COPPER	19
Brand#32	SMALL BURNISHED STEEL	23
Brand#32	STANDARD BURNISHED STEEL	45
Brand#34	ECONOMY ANODIZED NICKEL	49
Brand#34	LARGE BURNISHED TIN	49
Brand#34	PROMO ANODIZED TIN	3
Brand#34	SMALL BRUSHED TIN	3
Brand#34	STANDARD BURNISHED TIN	23
Brand#35	MEDIUM BRUSHED STEEL	45
Brand#35	PROMO BURNISHED STEEL	14
Brand#35	SMALL BURNISHED STEEL	23
Brand#35	SMALL POLISHED COPPER	14
Brand#35	STANDARD PLATED COPPER	9
Brand#41	ECONOMY BRUSHED BRASS	23
Brand#41	LARGE BURNISHED STEEL	23
Brand#41	PROMO BURNISHED TIN	14
Brand#41	PROMO PLATED STEEL	36
Brand#41	PROMO POLISHED TIN	19
Brand#41	SMALL BURNISHED COPPER	23
Brand#42	LARGE POLISHED TIN	14
Brand#42	MEDIUM ANODIZED TIN	49
Brand#42	MEDIUM BRUSHED TIN	14
Brand#42	MEDIUM BURNISHED NICKEL	23
Brand#42	MEDIUM PLATED COPPER	45
Brand#42	MEDIUM PLATED TIN	45
Brand#42	SMALL PLATED COPPER	36
Brand#43	ECONOMY BRUSHED STEEL	45
Brand#43	LARGE BRUSHED COPPER	19
Brand#43	PROMO BRUSHED BRASS	36
Brand#43	SMALL BURNISHED TIN	45
Brand#43	SMALL PLATED COPPER	45
Brand#44	PROMO POLISHED TIN	23
Brand#44	SMALL POLISHED NICKEL	14
Brand#55	LARGE PLATED STEEL	9

..... ROWS REMOVED .....
.....

Brand#55	LARGE PLATED TIN	9
4.		
Brand#55	LARGE PLATED TIN	14
4.		
Brand#55	LARGE PLATED TIN	23
4.		
Brand#55	LARGE POLISHED NICKEL	3
4.		
Brand#55	LARGE POLISHED STEEL	36
4.		
Brand#55	LARGE POLISHED STEEL	45
4.		
Brand#55	MEDIUM ANODIZED COPPER	9
4.		
Brand#55	MEDIUM BRUSHED BRASS	3
4.		
Brand#55	MEDIUM BRUSHED NICKEL	23
4.		
Brand#55	MEDIUM BRUSHED TIN	45
4.		
Brand#55	MEDIUM BURNISHED BRASS	23
4.		
Brand#55	MEDIUM BURNISHED COPPER	36
4.		
Brand#55	MEDIUM BURNISHED NICKEL	3
4.		
Brand#55	MEDIUM BURNISHED STEEL	14
4.		
Brand#55	MEDIUM BURNISHED STEEL	36
4.		
Brand#55	MEDIUM PLATED NICKEL	23
4.		
Brand#55	PROMO ANODIZED COPPER	14
4.		
Brand#55	PROMO ANODIZED COPPER	49
4.		
Brand#55	PROMO ANODIZED STEEL	36
4.		
Brand#55	PROMO ANODIZED TIN	23
4.		
Brand#55	PROMO BRUSHED NICKEL	36
4.		
Brand#55	PROMO BRUSHED STEEL	3
4.		
Brand#55	PROMO BRUSHED STEEL	36
4.		
Brand#55	PROMO BRUSHED TIN	9
4.		
Brand#55	PROMO BURNISHED COPPER	3
4.		
Brand#55	PROMO BURNISHED STEEL	14
4.		
Brand#55	PROMO BURNISHED TIN	23
4.		
Brand#55	PROMO BURNISHED TIN	49
4.		
Brand#55	PROMO PLATED COPPER	3
4.		
Brand#55	PROMO PLATED NICKEL	3
4.		
Brand#55	PROMO PLATED NICKEL	14
4.		
Brand#55	PROMO PLATED NICKEL	23
4.		
Brand#55	PROMO PLATED TIN	3
4.		
Brand#55	PROMO POLISHED COPPER	3
4.		
Brand#55	SMALL ANODIZED BRASS	19
4.		
Brand#55	SMALL ANODIZED NICKEL	45
4.		
Brand#55	SMALL BRUSHED COPPER	14
4.		
Brand#55	SMALL BRUSHED COPPER	45
4.		
Brand#55	SMALL BURNISHED BRASS	14
4.		
Brand#55	SMALL BURNISHED TIN	3
4.		
Brand#55	SMALL BURNISHED TIN	49
4.		
Brand#55	SMALL PLATED BRASS	45
4.		
Brand#55	SMALL PLATED COPPER	23
4.		
Brand#55	SMALL PLATED COPPER	36
4.		
Brand#55	SMALL PLATED COPPER	45
4.		
Brand#55	SMALL PLATED COPPER	49
4.		
Brand#55	SMALL PLATED NICKEL	9
4.		
Brand#55	SMALL PLATED STEEL	9
4.		
Brand#55	SMALL PLATED TIN	14
4.		
Brand#55	SMALL PLATED TIN	36
4.		
Brand#55	SMALL POLISHED NICKEL	45
4.		
Brand#55	SMALL POLISHED STEEL	19
4.		
Brand#55	SMALL POLISHED TIN	19
4.		
Brand#55	STANDARD ANODIZED BRASS	36
4.		
Brand#55	STANDARD ANODIZED BRASS	49
4.		
Brand#55	STANDARD ANODIZED STEEL	19
4.		
Brand#55	STANDARD ANODIZED TIN	36
4.		
Brand#55	STANDARD ANODIZED TIN	49
4.		
Brand#55	STANDARD BRUSHED BRASS	36
4.		
Brand#55	STANDARD BRUSHED COPPER	3
4.		
Brand#55	STANDARD BRUSHED COPPER	9
4.		
Brand#55	STANDARD BRUSHED COPPER	23
4.		
Brand#55	STANDARD BRUSHED STEEL	19
4.		
Brand#55	STANDARD BRUSHED TIN	23
4.		
Brand#55	STANDARD BRUSHED TIN	45
4.		
Brand#55	STANDARD BURNISHED BRASS	19
4.		
Brand#55	STANDARD BURNISHED NICKEL	3
4.		
Brand#55	STANDARD BURNISHED NICKEL	36
4.		
Brand#55	STANDARD BURNISHED STEEL	19
4.		
Brand#55	STANDARD PLATED BRASS	23
4.		
Brand#55	STANDARD PLATED NICKEL	9
4.		

```
Brand#55    STANDARD PLATED TIN              36
 4.
Brand#55    STANDARD POLISHED BRASS           3
 4.
Brand#55    STANDARD POLISHED BRASS          49
 4.
Brand#55    STANDARD POLISHED COPPER         19
 4.
Brand#55    STANDARD POLISHED COPPER         36
 4.
Brand#55    STANDARD POLISHED NICKEL         14
 4.
Brand#55    STANDARD POLISHED STEEL          9
 4.
Brand#55    STANDARD POLISHED STEEL          36
 4.
Brand#11    LARGE ANODIZED TIN               45
 3.
Brand#12    LARGE BURNISHED NICKEL           14
 3.
Brand#12    MEDIUM PLATED TIN                49
 3.
Brand#12    PROMO POLISHED TIN               3
 3.
Brand#13    STANDARD POLISHED NICKEL         19
 3.
Brand#15    SMALL BURNISHED STEEL            9
 3.
Brand#21    MEDIUM ANODIZED TIN              9
 3.
Brand#22    PROMO BRUSHED BRASS              19
 3.
Brand#22    PROMO BURNISHED COPPER           14
 3.
Brand#22    STANDARD BURNISHED COPPER        45
 3.
Brand#23    SMALL ANODIZED BRASS             23
 3.
Brand#24    PROMO BRUSHED STEEL              49
 3.
Brand#31    SMALL PLATED NICKEL              36
 3.
Brand#32    LARGE BRUSHED STEEL              45
 3.
Brand#41    PROMO BURNISHED NICKEL           36
 3.
Brand#43    STANDARD BRUSHED BRASS           23
 3.
Brand#44    MEDIUM ANODIZED NICKEL           9
 3.
Brand#51    MEDIUM BRUSHED TIN               36
 3.
Brand#53    MEDIUM BURNISHED BRASS           49
 3.
Brand#54    SMALL POLISHED BRASS             9
 3.

Number of rows retrieved is:  2762
------------------------------------------------


Stop timestamp  05/01/98 13:54:46
*****************************
QUERY 17
*****************************
Start timestamp  05/01/98 13:54:50

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

Tag: Q17      Sequence number: 12

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
----------------------
        24436.880


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


Stop timestamp  05/01/98 13:54:50
```

C.2 Test Database Table Contents

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

R_REGIONKEY R_NAME                R_COMMENT


----------- ------------------------- ----------------------------------------
--------------------------------------------------------------------------------
------------------------------
          2 ASIA                  NSg6xlMlA1lzm6mOR0Ajx
nhRA77NgRxBwL1M6Py R
jySB3RLwkyPkwMM2R1BQ
xAzkOgkjmll0gAghinP5inmNmR76MlijMS3S2zxONR15

          3 EUROPE
zlSL7Qwg12hMBL5lhlz0M45QkjShwSyiO04MLOh7wn
1ARLQPyPAyAiil576l1Li7AlnR1S RQ4SLny7B2Ryj5P66MLhn
NxhwB4C3ig0SO

          4 MIDDLE EAST           RllxmhPLz3Cy2mNlg4QMBnNASM
ACki MPki7Oi


          0 AFRICA
xSx31zz31Cl1z4OAnmm05AjiOxC3AMMNOgC0kACgwn
gg3glP7LLLywlQy7R

          1 AMERICA
kgyh3LSnC72k6zlAz0LP3k2L4QB1QL1O673OjO1SPj
0ngQ7CO100SBgmgRQ4lgPCMk21A425iklyAR4yBRAwR4Cm5miNw
4jl13mMnxw17B


  5 record(s) selected.


SELECT * FROM TPCD.NATION FETCH FIRST 10 ROWS ONLY

N_NATIONKEY N_NAME                N_REGIONKEY N_COMMENT


----------- ------------------------- ----------- ------------------------------
--------------------------------------------------------------------------------
----------------------------------------
          2 BRAZIL                            1
gLmS0nACAmnBCj2klki7RCPNgPxnCO
jNg4k OiAg57COSOm1NwCnOyLx40R SC
y20gPPAkNk5hxRhR5OmgS1iPQQzNAxPL30n67OgyC l617S
h4LS
```

```
         7 GERMANY                     3 z nOP4RkwO CmzBB 516mAg
lByw4O
M3QyNPA

        10 IRAN                        4
h532g43BgShyO50OgSB2hO6jxQn3Q1
w4NmPL4hg0mmPOC5SSLg6miR7m6B317 lz4jQMwPRyRn1jiRPmi
mk0 yj0C5M5xi

        13 JORDAN                      4 P53ljBPS
2PM7g6MwwywM47mCji1Qk
B4RNghzxB56mC5QiwOn2QlM3ikz6wN5NlM4N7C7B

        16 MOZAMBIQUE                  0
LB6SkSOmkznRLS4z5P2BOmC23RnA6h
7mn0gPAAN7nkxy6jO0k3w3R32R6A1ASk0LMyngO36jCL1gl5wQw4A
MPC

        20 SAUDI ARABIA                4 kC7 nNNz3ON73Az03O
S13CklRh47
jlxw5 5k6LClx52LQCQS6P1B3k1xPgMnk3PmMy7
l1nlBBBwBk0x6MACxOiigLO5mm5mO7N

        24 UNITED STATES               1
QQ41AxzBSQACRA1wCLyONCi5yzCylj
QR144lnNLm2 Q7PRyk BRzP

         3 CANADA                      1 4yMO AhnQ5Lh
wzQAM662Aw1ByCl7C
xmzRwNR5nAlO4 x

         6 FRANCE                      3 3mjmizl S
3L3k2hNNhNlP4w370xRx
yN15wn

        12 JAPAN                       2 y
PiRn6O4Rny0P6Q0NhM5N26AhghiC
lRQlP QNQx205B3im7zPBOSSm5MAl3BjMLiM3P


  10 record(s) selected.


SELECT * FROM TPCD.PART P_PARTKEY FETCH FIRST 10 ROWS
ONLY

P_PARTKEY   P_NAME                                        P_MFGR
          P_BRAND    P_TYPE                 P_SIZE      P_CONTAINER
P_RET
AILPRICE          P_COMMENT
----------- ---------------------------------------------------- ------------
------------- ---------- ------------------------- ----------- ----------- -----
-------------------- -----------------------
      1468 floral indian cream antique magenta                  Manufacturer
#2          Brand#22   ECONOMY BRUSHED NICKEL           27 SM
JAR       +1.
36946000000000E+003 lBgzOg0Sg1lBRQyyx25h
      1570 chiffon maroon rosy thistle                          Manufacturer
#1          Brand#15   LARGE BURNISHED STEEL            41 LG CASE
+1.
47157000000000E+003 mAC0BNOMO nwkl
      1600 frosted indian beige wheat smoke                     Manufacturer
#2          Brand#21   STANDARD PLATED COPPER           21 LG
DRUM      +1.
50160000000000E+003 R5Qx1A
      1675 rosy burlywood blush linen gainsboro                 Manufacturer
#3          Brand#32   SMALL POLISHED TIN               45 MED PACK
+1.
57667000000000E+003 h mgh21SzRiBgQCxhji0R
      1754 papaya pink light wheat burlywood                    Manufacturer
#1          Brand#11   LARGE BURNISHED NICKEL           39 LG
CASE      +1.
65575000000000E+003 xCyBBjnzA3407
      1950 indian cyan firebrick chartreuse                     Manufacturer
#5          Brand#51   MEDIUM BRUSHED BRASS             48 MED
CASE      +1.
85195000000000E+003 75PQ3R
      1964 purple dodger sky ivory olive                        Manufacturer
#2          Brand#25   STANDARD ANODIZED NICKEL          42 MED
JAR       +1.
86596000000000E+003 mC14RAOzP7g3Nm
      1981 cyan linen ghost saddle dodger                       Manufacturer
#4          Brand#44   MEDIUM BURNISHED NICKEL           23 SM
BAG       +1.
88298000000000E+003 x3O 0yiSj lR
      2044 ivory honeydew black lawn lavender                   Manufacturer
#4          Brand#41   SMALL PLATED COPPER              47 SM PKG
+9.
46040000000000E+002 xC13lS NOBjlL
      2089 steel moccasin dim cornflower orchid                 Manufacturer
#5          Brand#51   PROMO ANODIZED COPPER            33 SM PKG
+9.
91080000000000E+002 66M2inA145N2A1NL

  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

----------- ------------------------- ---------------------------------------- -
---------- --------------- ------------------------- ----------------------------
-------------------------------------------------------------------------------
         997 Supplier#000000997        0POP6iz5Om1xR4zyi7Q
SB22hli2L7i2C1gOCPm
          3 13-221-322-7971   +3.65956000000000E+003
xjxyAyNmByiS7z6gyPi70lzn02Aj
Rw y lz6miP3SmmyAn742QySQnBSRkhljSCPL0QSmhzLLSN
hN62g3RlP5k5l1CxNj
         998 Supplier#000000998        w6zy4w0Mg5L76nR4 OQ3hwPgk5
         15 25-430-605-1180   +3.28262000000000E+003
yhz5LLCgyBg7PNjmAOjQMB34QBP6
A
      1008 Supplier#000001008        x26xLx5nBj SQh
nmRBySyRgmhhMjgnyxk5L7
         19 29-218-505-4622   -8.58860000000000E+002
j0m7CM46kBjz1wkwROSlgPMC4OlA
03QOiC67RBPAkn
      1061 Supplier#000001061        63gzS g7 S
         10 20-204-356-2184   +3.75159000000000E+003
2z27km5lzlCN2nxlx RyQ2mRR5nm
LBQy30z Q34 5MM
      1111 Supplier#000001111        LySj2hxCPgl3wz2jSl47nmly5Nz
         13 23-375-309-6302   +6.38792000000000E+003
m4nOAwxjC7xjOgzjPQx6wl22Sn g
4
      1201 Supplier#000001201        7hhjkCA1k0mC
         21 31-700-109-8793   +2.32692000000000E+003 Ann6kNw0g
NRlzQOjNxkm6SLlMh2
67gjjhSOMhgR6 g262QxxkNO4OL0i523NN3igBOgL4B0N2SNwBOO1y
      1208 Supplier#000001208        6 N4jn1CgwMSz
2RjlmA70BR3jOh377jCgjM720w
          6 16-739-665-8270   +5.00757000000000E+003
A4ikQxNClxNy5im1zCiQ43Q7h


      1229 Supplier#000001229        i2zBmiQlji6xNSQ1igz2zOR
         23 33-448-290-4429   +4.37967000000000E+003
y2BAyxy6Rklkm5ALk1RLi 3CCMN2
iRw2ngl67BOOxnglN5  mx33jQBLlMQz4wh6nxOSlwiwnyPA1
      1281 Supplier#000001281
1QOhLnSSyLNM6NLRNmiLhRMQSwNNm
```

```
      23 33-432-130-8820   +5.79191000000000E+003
x1ghCw01lMSmQhCgPgPPOLk2yCj6
Og4RN2MMyw72lLww74 7jz
gCnBwm6PgPhLgiRliRAQ3MzjhL7RzB0yzNikP5NhzSA
     1282 Supplier#000001282        wBS QBLhNSQ2 2BNnSP2
Nm707Bhyhhy7wPSB
       2 12-744-273-6433   +3.65360000000000E+003
3wnmg44OmiNlxkhgjOgh0CSw0jP1
 jM3hMkjPx4BSMyAMlNRgBS3Szh47N5w1iO3jPjgxLCPkCLQQkjMM

 10 record(s) selected.


SELECT * FROM TPCD.PARTSUPP FETCH FIRST 10 ROWS ONLY

PS_PARTKEY  PS_SUPPKEY  PS_AVAILQTY PS_SUPPLYCOST
PS_COMMENT



----------- ----------- ----------- ------------------------ --------------------
--------------------------------------------------------------------------------
--------------------------------------------------------------------------------
--------------------
       89          90        3430   +7.44870000000000E+002 26Lyjk zh
1My0k7g20
PP2Q7A6 2kgg203gni0nPgLwAN73 iN 7h625L37iNSi3PlgxmjLCN7
kSNz7nB7Cmnik215nRlP12mz
R4zRPhR0AM3nmk063LigNmMlhyz5kN5C hBQ0i3hmhwnlg
LC4P3ySh2L0A67PgjSx274 M

       89     2500090        8599   +7.76530000000000E+002 BxOLxQL
0Myk5n6mzMR
n34 CMB27Qyh2 hS1Q66i1Am1zSy7gLlRQ2NLkLwx 4 0NNN2Mmz
k1SOg1RkyyS5B mShwR0h0 MOzw
i5xOxNz4wAx4Nx1PLn6 myBmNNjg5jjQ6 m6lyOz7ARgw

       89     5000090        7876   +4.17610000000000E+002 x1C5RN
RSjCPwQl C6O
LlMl4l1N3Mm126B6Oj 7k4k1g70S mhR67y RBlLMQ6
N1g5MkBNxSl4wkk2LL0yj102CQQhRLQ1wOA7
Ow6C5CnS5A0hPQSM40lz5xC

       89     7500090         924   +9.20020000000000E+002 m Q
gQlLk7hn6P0i06z
m3
hR6l7LRmxPwBMgMRx6RyNmAgkjO3gNlzRwPRACBiAkn0z52SkSkk
B5A6OzMO5B1Q0jSMR1zCR6R6B
hA450jmRi Bn4l2N3wgL77h6ynO2Rjw3wRlOShhjCygl 3 1hzC5wRxSg

       126        127        2647   +2.21890000000000E+002 x3C
Bkw145izSP4Qigw
4yNRmyM3B3Mwn70nw5mCnN3Q5OShkMS3NiAL5210iPQklg hO
Q70150QkgSNx4jRAmAjwS nBn56wil
L4AS6khzkyLPyhLlhxnwARQ26AQ2w1 NCB66 4n0Lh

       126     2500127        2373   +1.94380000000000E+002
7R4Rm0hQCz5igRxkBA3
07g 6B1xBCCCg0xB4CC3A01QC6yzz N07N



       126     5000127        1532   +4.51610000000000E+002 w3m5 BkBi
A6mjBNBm1
ny0x7h56NAMNQh6Mnw5513MRR
kgyli4kRC2nA2l43NQhMm50LhxSgB4n6n4QzwMB1l7g06nQNSgL7NR
g5zlNLR6j6k7AlhglPQiMM5xzSNNznQN40izzANN4
5mSmzk2yw2Ow7S75SQ5n2iPOklx mLnLOm7CmP
mlj zk6kC2Q
       126     7500127        5458   +9.29430000000000E+002
L1RhA1yNShzS45O6B37
gn45MxgBC2lwjh24nNB3hBlxOygP3Awzl55hn2kMB1n4BwML2h6Nyg
ihjhlLOMnNxzOO2OlOLjCRhx2
g4yjxNCmxxQQmLy3iMh1h3SMm75yLlgRm7iz04SR27CMA1n4xlQ
BPSlLBS6z7MQ5mz7C35P

       169        170        6914   +6.19530000000000E+002 0ng
BQC4AlnnihxPi3
PBCOLAizkkxAlMM4jLR3nzR1gyzMg64 Ag4mL 556O7n0L7RQ
4OnBhCnkj OxSC5SAlM1BiC


       169     2500170        6589   +9.47030000000000E+002
wPCl5k4wAxmC 0 6m54
zBm 4mg17nR7NM z6ilN
ynwziwMnynmMAhOBzmS03ngA0BiizSRkMg00wyy 6A6wLS 5lnN0P0



 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



----------- ------------------------ ---------------------------------------- -
---------- --------------- ------------------------ ------------ ---------------
--------------------------------------------------------------------------------
--------------------
       42 Customer#000000042        MNQx34mh32mA
       5 15-416-330-4175   +8.72701000000000E+003 BUILDING
R2B2mi67R30nn2S
wzMMB02n4624AM3zALkhikxgLzkwOjxi66OMN0mOkg3nNmglAhmC
6BOgB

       62 Customer#000000062        P7MglgmS3j k0
       7 17-361-978-7059   +5.95610000000000E+002 MACHINERY
wiClwz0nx4BQORA
PlnkQ2Alm NSzlmzjygAjiLg6

      128 Customer#000000128        LynliQ5iCS5lRS y1
       4 14-280-874-8044   -9.86960000000000E+002 HOUSEHOLD
C1AMLwROS103myk
5xyLkmPLh430m45mw5B 4BgL5nQBRPhxP41 62MMk Cn34PzL
g3CA4yQPmRCL5NMSPnOB7PMQPgL3yS
hwwwhP
       376 Customer#000000376        x4N0MmhjRP5yCR40jQPOihOA
       16 26-437-952-8986   +4.23145000000000E+003 AUTOMOBILE   j
nyA0Qijh4QMyN
5k44LmhN0PxMOyCnklgP4Myk355Q6Qg3mR1gni

       434 Customer#000000434        Am5x12nyM g62n5kl11k5nhNA
       3 13-325-443-1474   +2.94046000000000E+003 MACHINERY
gSy6nhPNA 1h0iy
BlxLOmlig lQA47RA3gkiLk4iAiwj3Cynig00kQPOinLh6AwyklBxhCy3z
4Nl6jiBn0BlC664LNP4LL
khnO53m5N1My1CwAOBOQx
       457 Customer#000000457        i15gx1hm j2
gx71mhiQ3jxx1l17CR7jgNMyOy3
       20 30-543-684-2857   +5.86761000000000E+003 FURNITURE
SM6LwS3iiii00j5
Q60BL y4P 0mB1Pz6hP7lmM45OzkL0kmm02zCL5niM0zxCPSSxlQ5hA

       462 Customer#000000462        7yQnSASwgy5mLQ LjQygh6
       21 31-157-561-4106   +4.52260000000000E+003 HOUSEHOLD    7
hB3nB4 hS7ywL
0lyB5lOBAwwPQR1g5nQR574MzO0j

       471 Customer#000000471        A5mzhh5n3CC50gM5gn7
       4 14-574-118-1005   +5.71690000000000E+003 FURNITURE
25SS ylS5n35 Oi
```

```
 zOl4zQm6CO4lA44Ok1CMl3NAn1QzOxi57k40nR75
5Sxlxz6RwNAiwCQg2

      504 Customer#000000504       NP6P7 g3
Ml0y0jMBN66343xS0j61xCy7lnAm 0g
      10 20-916-264-7594   +5.10000000000000E-001 FURNITURE
Aykgn5hyNzj0hQg
4n2Q0O6CnlCwjSzLhzNBily6AOCi7Amgxgz04zNkkLwBjmRwPjNlOSL
jgL 6N jljikQPw2LQR7CjQli
n1 Ngyzl1
      522 Customer#000000522       QSmNkmgMAx 0jyNl2A3B
wLjB6j2RC7g
      12 22-771-454-9561   +6.35846000000000E+003 BUILDING
5zyj6hPizm2MLi3
PPCi2wlkgwx5hMNOOlRkR


  10 record(s) selected.


select char(o_orderkey) as o_orderkey, o_custkey, o_orderstatus,
o_totalprice, o
_orderdate, o_orderpriority, o_clerk, o_shippriority, o_comment from
tpcd.orders
 FETCH FIRST 10 ROWS ONLY

O_ORDERKEY           O_CUSTKEY   O_ORDERSTATUS
O_TOTALPRICE         O_ORDERD
ATE O_ORDERPRIORITY O_CLERK        O_SHIPPRIORITY
O_COMMENT

-------------------- ----------- -------------- ------------------------- --------
--- --------------- --------------- --------------- -----------------------------
-----------------------------------------------
32                   127587628 O              +2.27117010000000E+005
07/16/19
95  2-HIGH           Clerk#000615602              0
7ihNSz00NCxA31PPx6RM4ih BPPlz
417SLk3SRA1zx0nlikRgjkx
615                   75810850 F              +1.60212300000000E+004
05/09/19
92  5-LOW            Clerk#000387603              0
S0kmS2iAhMhN4z6ni1OQ5O7LxQkwl
Nx0hQi
642                  102677749 F              +3.88338200000000E+004
12/16/19
93  3-MEDIUM         Clerk#000356123              0
Cx0yA1giLN1gmm7xSBANBnP1Nwn4Q
kn4B2lAg LQQnQRQN7ij3wC
645                  146879158 F              +3.39796560000000E+005
12/03/19
94  2-HIGH           Clerk#000089321              0
4C1OhPx55CAgSPRS3Bx2741wiNBw5
CNB0zA5jwP44lw2hO
678                  119525875 F              +2.24145300000000E+005
02/27/19
93  5-LOW            Clerk#000529295              0 iMS22 zy30MSjQ2l
lAN7wPNn2RBi
g3yz3CC3Sk i61Ci11L6lNiC4ghw2gMm N
896                  136236470 F              +3.27519240000000E+005
03/09/19
93  1-URGENT         Clerk#000186608              0
2z21m31CkwORLRh6QP4i1h4zigQCy
nAB52NMz 2jCR14xjPnzxyOlLwjBOO4i6M0nM10jz6Q4
961                   75489367 O              +2.52653230000000E+005
06/04/19
95  4-NOT SPECIFIED Clerk#000719609              0
7w3BMBA5y0QjhxSCj1ySRRnB75gn
BSxMR7w4i1M1jNwP0hlAzAkyOwjAmjR7
966                  101202617 O              +1.66533290000000E+005
04/30/19
98  2-HIGH           Clerk#000238482              0
PxM2hPCOhgngBxlLLg5y3l

1188                 60031427 O               +1.20360080000000E+005
04/11/19
96  2-HIGH           Clerk#000255563              0 BCR1z iyPh22yML Ahl0l

1281                 86110283 F               +1.64262760000000E+005
12/11/19
94  1-URGENT         Clerk#000429891              0 PiQCL1zPQ
PmNmOl47N0LS4m0NhA
h14mg1274w1A


  10 record(s) selected.


select char(l_orderkey) as l_orderkey, l_partkey, l_suppkey, l_linenumber,
l_qua
ntity, l_extendedprice, l_discount, l_tax, l_returnflag, l_linestatus,
l_shipdat
e, l_commitdate, l_receiptdate, l_shipinstruct, l_shipmode, l_comment
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
-------------------- ----------- ----------- ------------ ----------------------
-- ------------------------- ------------------------ ------------------------ --
---------- ------------ ---------- ------------ ------------- ------------------
------- ---------- ----------------------------------------
97                    36880454     1880461           1   +1.80000000000000E+0
01   +2.57869800000000E+004   +9.00000000000000E-002
+2.00000000000000E-002 R
        F          04/28/1993 04/06/1993   05/04/1993    DELIVER IN
PERSON
      MAIL       00hwxhzxxlnBnS B L3m7h5
97                    49778960     9778961           2   +2.60000000000000E+0
01   +5.29484800000000E+004   +3.00000000000000E-002
+3.00000000000000E-002 R
        F          03/27/1993 04/28/1993   03/31/1993    COLLECT COD
      AIR        NhMABjyMCQijgygAP1Q nMlx6mjCQ2365 5x2
97                    34977866     9977873           3   +1.20000000000000E+0
01   +2.33054400000000E+004   +4.00000000000000E-002
+8.00000000000000E-002 A
        F          05/11/1993 03/12/1993   06/06/1993    COLLECT COD
      MAIL       wy6RmzzQ43iyhyzR3jwj5RMyxMQ7M
451                   79632642     2132650           1   +1.10000000000000E+0
01   +1.72772600000000E+004   +0.00000000000000E+000
+1.00000000000000E-002 N
        O          07/29/1998 07/27/1998   08/11/1998    DELIVER IN
PERSON
      FOB        l0LNQ ygQy4l6gBAlNizCxQ5Lhw14236nCBL33n3zy
451                   185797953     8297972           2
+4.00000000000000E+0
01   +8.16668000000000E+004   +1.00000000000000E-001
+3.00000000000000E-002 N
        O          08/04/1998 08/10/1998   08/22/1998    TAKE BACK
RETURN
      FOB        lCM3AN6A7gP6kxRPnk0CzALiiRMOlP0
451                   106193617     3693648           3
+8.00000000000000E+0
00   +1.36424800000000E+004   +0.00000000000000E+000
+7.00000000000000E-002 N
        O          06/16/1998 08/15/1998   07/06/1998    NONE
      REG AIR    hx37y2PC6zMN5MkjRyRQ3OwMRl3mkM
451                   196118389     3618447           4
+4.00000000000000E+0
```

01 +5.59032000000000E+004 +1.00000000000000E-001
+1.00000000000000E-002 N
O 08/07/1998 07/20/1998 08/30/1998 DELIVER IN
PERSON
FOB l4jLA6Mknznl
834 72939825 439847 1 +1.50000000000000E+0
01 +2.79177000000000E+004 +1.00000000000000E-002
+2.00000000000000E-002 R
F 07/08/1994 07/22/1994 07/28/1994 DELIVER IN
PERSON
FOB l6k5PO5l2wh7QA ANkkz0
834 99637190 7137218 2 +2.00000000000000E+0
01 +2.24442000000000E+004 +6.00000000000000E-002
+2.00000000000000E-002 A
F 06/29/1994 07/05/1994 07/14/1994 COLLECT COD
REG AIR CyM7Q0n5PMhMggNj3Mj17n 441xx147jgx
931 117651130 151142 1 +4.60000000000000E+0
01 +4.94615000000000E+004 +0.00000000000000E+000
+4.00000000000000E-002 A
F 03/17/1993 01/31/1993 03/29/1993 COLLECT COD
FOB mw0mABC20S

10 record(s) selected.

C.3 Query Substitution Parameters

Power stream Seed = 225520782
-- TPC-D Parameter Substitution (Version 1.3.1)
-- using 225520782 as a seed to the RNG
Q1 DELTA 116
Q2 SIZE 47
TYPE TIN
REGION MIDDLE EAST
Q3 SEGMENT MACHINERY
DATE 1995-03-20
Q4 DATE 1997-07-01
Q5 REGION MIDDLE EAST
DATE 1996-01-01
Q6 DATE 1997-01-01
DISCOUNT 0.07
QUANTITY 25
Q7 NATION1 UNITED KINGDOM
NATION2 MOROCCO
Q8 NATION UNITED KINGDOM
REGION EUROPE
TYPE LARGE BRUSHED TIN
Q9 COLOR thistle
Q10 DATE 1994-12-01
Q11 NATION UNITED KINGDOM
FRACTION 0.0000001000
Q12 SHIPMODE1 SHIP
SHIPMODE2 MAIL
DATE 1997-01-01
Q13 CLERK Clerk#000000927
Q14 DATE 1997-09-01
Q15 DATE 1997-07-01
Q16 BRAND Brand#54
TYPE ECONOMY BRUSHED
SIZE1 33
SIZE2 42
SIZE3 21
SIZE4 34
SIZE5 38
SIZE6 47
SIZE7 49
SIZE8 20
Q17 BRAND Brand#54
CONTAINER WRAP CASE
Throughput Stream = 1 Seed = 1367916679
-- TPC-D Parameter Substitution (Version 1.3.1)
-- using 1367916679 as a seed to the RNG
Q1 DELTA 120
Q2 SIZE 50
TYPE NICKEL
REGION MIDDLE EAST
Q3 SEGMENT MACHINERY
DATE 1995-03-06
Q4 DATE 1997-11-01
Q5 REGION MIDDLE EAST
DATE 1993-01-01
Q6 DATE 1997-01-01
DISCOUNT 0.03
QUANTITY 25
Q7 NATION1 UNITED STATES
NATION2 EGYPT
Q8 NATION UNITED STATES
REGION AMERICA
TYPE SMALL ANODIZED NICKEL
Q9 COLOR yellow
Q10 DATE 1995-01-01
Q11 NATION UNITED STATES
FRACTION 0.0000001000
Q12 SHIPMODE1 SHIP
SHIPMODE2 AIR
DATE 1997-01-01
Q13 CLERK Clerk#000000999
Q14 DATE 1998-01-01
Q15 DATE 1997-11-01
Q16 BRAND Brand#51
TYPE PROMO BRUSHED
SIZE1 27
SIZE2 6
SIZE3 12
SIZE4 35
SIZE5 31
SIZE6 14
SIZE7 17
SIZE8 29
Q17 BRAND Brand#51
CONTAINER WRAP DRUM
Throughput Stream = 2 Seed = 1916419193
-- TPC-D Parameter Substitution (Version 1.3.1)
-- using 1916419193 as a seed to the RNG
Q1 DELTA 69
Q2 SIZE 9
TYPE NICKEL
REGION MIDDLE EAST
Q3 SEGMENT AUTOMOBILE
DATE 1995-03-14
Q4 DATE 1993-11-01
Q5 REGION AFRICA
DATE 1995-01-01
Q6 DATE 1993-01-01
DISCOUNT 0.05
QUANTITY 25
Q7 NATION1 EGYPT
NATION2 IRAQ
Q8 NATION EGYPT
REGION MIDDLE EAST
TYPE MEDIUM POLISHED NICKEL
Q9 COLOR chiffon
Q10 DATE 1993-05-01
Q11 NATION EGYPT
FRACTION 0.0000001000
Q12 SHIPMODE1 AIR
SHIPMODE2 TRUCK
DATE 1997-01-01
Q13 CLERK Clerk#000000163
Q14 DATE 1993-11-01
Q15 DATE 1993-11-01
Q16 BRAND Brand#13
TYPE PROMO ANODIZED
SIZE1 45
SIZE2 18

SIZE3 43
SIZE4 30
SIZE5 1
SIZE6 8
SIZE7 25
SIZE8 3
Q17 BRAND Brand#13
CONTAINER WRAP BAG
Throughput Stream = 3 Seed = 1175406391
-- TPC-D Parameter Substitution (Version 1.3.1)
-- using 1175406391 as a seed to the RNG
Q1 DELTA 119
Q2 SIZE 49
TYPE COPPER
REGION MIDDLE EAST
Q3 SEGMENT MACHINERY
DATE 1995-03-18
Q4 DATE 1997-10-01
Q5 REGION MIDDLE EAST
DATE 1995-01-01
Q6 DATE 1997-01-01
DISCOUNT 0.06
QUANTITY 25
Q7 NATION1 UNITED STATES
NATION2 KENYA
Q8 NATION UNITED STATES
REGION AMERICA
TYPE LARGE BURNISHED COPPER
Q9 COLOR wheat
Q10 DATE 1995-01-01
Q11 NATION UNITED STATES
FRACTION 0.0000001000
Q12 SHIPMODE1 SHIP
SHIPMODE2 TRUCK
DATE 1997-01-01
Q13 CLERK Clerk#000000979
Q14 DATE 1997-12-01
Q15 DATE 1997-10-01
Q16 BRAND Brand#53
TYPE ECONOMY BRUSHED
SIZE1 41
SIZE2 26
SIZE3 43
SIZE4 16
SIZE5 27
SIZE6 34
SIZE7 29
SIZE8 19
Q17 BRAND Brand#53
CONTAINER WRAP BOX
Throughput Stream = 4 Seed = 1869157532
-- TPC-D Parameter Substitution (Version 1.3.1)
-- using 1869157532 as a seed to the RNG
Q1 DELTA 79
Q2 SIZE 17
TYPE TIN
REGION EUROPE
Q3 SEGMENT BUILDING
DATE 1995-03-23
Q4 DATE 1994-08-01
Q5 REGION AMERICA
DATE 1996-01-01
Q6 DATE 1994-01-01
DISCOUNT 0.07
QUANTITY 25
Q7 NATION1 INDIA
NATION2 CHINA
Q8 NATION INDIA
REGION ASIA
TYPE ECONOMY PLATED TIN
Q9 COLOR frosted
Q10 DATE 1993-09-01
Q11 NATION INDIA
FRACTION 0.0000001000
Q12 SHIPMODE1 RAIL
SHIPMODE2 FOB
DATE 1996-01-01
Q13 CLERK Clerk#000000326
Q14 DATE 1994-09-01
Q15 DATE 1994-08-01
Q16 BRAND Brand#24
TYPE ECONOMY PLATED
SIZE1 22
SIZE2 28
SIZE3 34
SIZE4 5
SIZE5 26
SIZE6 40
SIZE7 41
SIZE8 4
Q17 BRAND Brand#24
CONTAINER JUMBO PKG
Throughput Stream = 5 Seed = 536650117
-- TPC-D Parameter Substitution (Version 1.3.1)
-- using 536650117 as a seed to the RNG
Q1 DELTA 118
Q2 SIZE 49
TYPE COPPER
REGION EUROPE
Q3 SEGMENT MACHINERY
DATE 1995-03-25
Q4 DATE 1997-09-01
Q5 REGION MIDDLE EAST
DATE 1996-01-01
Q6 DATE 1997-01-01
DISCOUNT 0.08
QUANTITY 25
Q7 NATION1 UNITED STATES
NATION2 ROMANIA
Q8 NATION UNITED STATES
REGION AMERICA
TYPE ECONOMY POLISHED COPPER
Q9 COLOR violet
Q10 DATE 1995-01-01
Q11 NATION UNITED STATES
FRACTION 0.0000001000
Q12 SHIPMODE1 SHIP
SHIPMODE2 FOB
DATE 1996-01-01
Q13 CLERK Clerk#000000962
Q14 DATE 1997-11-01
Q15 DATE 1997-09-01
Q16 BRAND Brand#54
TYPE ECONOMY PLATED
SIZE1 34
SIZE2 49
SIZE3 38
SIZE4 43
SIZE5 11
SIZE6 8
SIZE7 36
SIZE8 37
Q17 BRAND Brand#54
CONTAINER JUMBO PKG
Throughput Stream = 6 Seed = 722405354
-- TPC-D Parameter Substitution (Version 1.3.1)
-- using 722405354 as a seed to the RNG
Q1 DELTA 91
Q2 SIZE 26
TYPE COPPER
REGION AFRICA
Q3 SEGMENT FURNITURE
DATE 1995-03-20
Q4 DATE 1995-07-01

Q5 REGION ASIA
DATE 1996-01-01
Q6 DATE 1995-01-01
DISCOUNT 0.07
QUANTITY 24
Q7 NATION1 JAPAN
NATION2 MOROCCO
Q8 NATION JAPAN
REGION ASIA
TYPE LARGE POLISHED COPPER
Q9 COLOR magenta
Q10 DATE 1994-02-01
Q11 NATION JAPAN
FRACTION 0.0000001000
Q12 SHIPMODE1 TRUCK
SHIPMODE2 MAIL
DATE 1993-01-01
Q13 CLERK Clerk#000000511
Q14 DATE 1995-08-01
Q15 DATE 1995-07-01
Q16 BRAND Brand#34
TYPE STANDARD POLISHED
SIZE1 27
SIZE2 19
SIZE3 3
SIZE4 7
SIZE5 2
SIZE6 38
SIZE7 20
SIZE8 35
Q17 BRAND Brand#34
CONTAINER SM PACK
Throughput Stream = 7 Seed = 598060941
-- TPC-D Parameter Substitution (Version 1.3.1)
-- using 598060941 as a seed to the RNG
Q1 DELTA 60
Q2 SIZE 1
TYPE NICKEL
REGION AMERICA
Q3 SEGMENT AUTOMOBILE
DATE 1995-03-14
Q4 DATE 1993-02-01
Q5 REGION AFRICA
DATE 1995-01-01
Q6 DATE 1993-01-01
DISCOUNT 0.05
QUANTITY 24
Q7 NATION1 ALGERIA
NATION2 IRAQ
Q8 NATION ALGERIA
REGION AFRICA
TYPE MEDIUM POLISHED NICKEL
Q9 COLOR antique
Q10 DATE 1993-02-01
Q11 NATION ALGERIA
FRACTION 0.0000001000
Q12 SHIPMODE1 REG AIR
SHIPMODE2 TRUCK
DATE 1994-01-01
Q13 CLERK Clerk#000000016
Q14 DATE 1993-02-01
Q15 DATE 1993-02-01
Q16 BRAND Brand#13
TYPE MEDIUM BURNISHED
SIZE1 35
SIZE2 45
SIZE3 17
SIZE4 13
SIZE5 42
SIZE6 48
SIZE7 39
SIZE8 4
Q17 BRAND Brand#13
CONTAINER LG DRUM

Appendix D Driver Source Code

D.1 tpcdbatch.sqc

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

/** Necessary header files **/

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

#include <time.h>
#include <ctype.h>
#ifndef SQLWINT
#include <sys/time.h>                          /*@d33143aha*/
#include <sys/ipc.h>
#include <sys/sem.h>
#include <sys/mode.h>
#include <sys/timeb.h>
#include <sys/types.h>
#else
#include <windows.h>
#include <sys\timeb.h>
#endif
#include <errno.h>


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

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


/************************************************************
***********/
/* Define synonyms here                                     */
/************************************************************
***********/
#define TPCDBATCH_VERSION "3.0"

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

#define TPCDBATCH_MAX_COLS    100                   /* @d30369
tjg */

#define TPCDBATCH_CHAR char

#define TPCDBATCH_PRINT_FLOAT_WIDTH  20
  /* kmw - allow 15 whole digit for %#.3f format */
  /*     - note: use > 18, size of long indentifier so that it will */
  /*       be larger than any column heading                        */
#define TPCDBATCH_PRINT_FLOAT_MAX    1e15  /* kmw */
/* #define TPCD_PREPARETIME 1 */      /* for separate prep/exec on uf
jen 1106 */
#define UF2DEBUG 1

#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
#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


#define BLANKS      "\0"
#define READMODE    "r\0"
#define WRITEMODE   "w\0"
#define APPENDMODE  "a\0"
#define mem_error(xx)                                  \
{ fprintf(stderr,"\n--Out of memory when %s.\n",xx); }
/* Display out-of-memory and end */

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



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

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

/** Timer-necessary defines for portability **/
#if (defined (SQLOS2) || defined(SQLWINT)) || defined(SQLWIN) ||
defined(SQLDOS)
typedef struct timeb Timer_struct;
#elif (defined (SQLUNIX) || defined(SQLAIX))                /*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 5 /* maximum retries for deadlock encounters */

/*************************************************************
***************/
/* 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    */
  time_t               stream_start_time;    /* start time for stream       */
  time_t               stream_end_time;      /* end time for stream         */
  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                  sem_on;               /* semaphore stuff             */
  int                  copy_on_load;         /* indication of whether or not */
                                             /* to do use a copy directory   */
                                             /* (equiv to COPY YES) on load  */
                                             /* default is FALSE */
  long                 lSeed;                /* seed used to generate the    */
                                             /* queries for this particular  */
                                             /* run.                         */
  FILE                 *stream_list;         /* ptr to query list file      */
  char                 update_num_file[150]; /* name of file that keeps track */
                                             /* of which update pairs have run*/
  char                 sem_file[150];        /* semaphore name */
  FILE                 *stream_report_file;  /* file to report start stop */
                                             /* progress of the stream */
};

EXEC SQL BEGIN DECLARE SECTION;
 char set_query_stmt[25]="SET CURRENT DEGREE = '1'";
EXEC SQL END DECLARE SECTION;

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

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

};

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

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


/*************************************************************
***********/
/* Structure used to hold precision for decimal numbers               */
/*************************************************************
***********/
struct declen
{/* kmw */
  unsigned char m;        /* # of digits left of decimal */
  unsigned char n;        /* # of digits right of decimal */
};


/*************************************************************
***********/
/* Structure containing control flags passed between functions        */
/*************************************************************
***********/
struct ctrl_flags
{                                                    /* @d25594 tjg */
  int eo_infile;
  int time_stamp;
  int eo_block;                                      /* @d30369 tjg */
  int select_status;
};

EXEC SQL BEGIN DECLARE SECTION;
  char sourcefile[256];
EXEC SQL END DECLARE SECTION;

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

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

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

long error_check(void);                              /* @d28763 tjg */

void 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( int updatePair, int copyOnOrOff );
void runUF2( int updatePair, int copyOnOrOff );

#ifdef PARALLEL_UPDATES
void runUF1_fn( int updatePair, int i, int copyOnOrOff );
void runUF2_fn( int updatePair, int i, int deleteChunk, int copyOnOrOff );
#endif

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

char *get_time_stamp(int form, time_t time_pointer);      /* @d25594 tjg
*/
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);
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);


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";

EXEC SQL END DECLARE SECTION;


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


/** Other globals **/
FILE            *instream, *outstream;  /* File pointers            */
int             verbose = 0;            /* Verbose option flag      */
int             updatePairStart;        /* update pair to start at  */
int             currentUpdatePair;      /* update pair running      */
int             updatePairStop;         /* update pair to stop before */
char            newtime[50]="\0";       /* Des - moved from get_time_stamp
*/
char            outstreamfilename[256]; /* store filename of outstream
                                   wlc 081397 */
char            tempdir[256];           /* to hold TPCD_TMP_DIR
                                   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)
*/
```

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


  struct stmt_info     s_info = { -1, -1, 0, 1, -1, -1, "\0", "\0", "\0", NULL };
                                   /* first stmt_info structure */

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

  struct global_struct g_struct =
            { NULL, NULL, NULL, NULL, 0, 0, "\0", 0.1, "\0", 1, FALSE,
0,
              NULL, "\0", "\0", NULL };



/** 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 */

/************************************************************
******************
 *                                                          *
 *   This is the transition from the "driver" to the "SUT"                  *
 *                                                          *

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

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

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

    c_flags.eo_block = 0;

    if (c_l_opt.outfile)
      output_file(&g_struct); /* determine appropriate name for output
files */

    get_start_time(&start_time);
    strcpy(g_struct.s_info_ptr->start_stamp,
           get_time_stamp(T_STAMP_FORM_1,(time_t)NULL) );

    /* write the start timestamp to the file...if this is not a qualification */
    /* run, then write the seed used as well */
    fprintf( outstream,"Start timestamp %18.18s \n",
           g_struct.s_info_ptr->start_stamp);
    if (c_l_opt.intStreamNum >= 0)
    {
      if (g_struct.lSeed == -1)
      {
        fprintf( outstream,"Using default qgen seed file");
```

```
        }
        else
          fprintf( outstream,"Seed used = %ld",g_struct.lSeed);

        fprintf( outstream,"\n");
      }

      do {  /* Loop through these statements as long as we haven't reached
               the end of the input file or the end of a block of statements
            */

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

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

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

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

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

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


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


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


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

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

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

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

    return(0);

}  /* end of main  */
```

```
/*************************************************************
***********/
/* Generic form of Sleep */
int SleepSome( int amount)
{
#ifndef SQLWINT
  sleep (amount);
#else
  sleep (amount*1000);          /* 10x for NT */
#endif
  return;
}

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

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

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

  int good_sql = 0;              /* good-sql stmt flag   @d23684 tjg */
  int stmt_num_flag = 1;          /* first line of SQL stmt flag      */
  int eostmt = 0;               /* flag to signal end of statement  */


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

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

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

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

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

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

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

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

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

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

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

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

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

      eostmt=0;
    }

    /* Need this hack here to check if there's any more empty lines left
       in the input file.  Continue only if there are aren't any */
    else if (strcmp(input_ln, "\0")) /* HACK */ {   /* A regular SQL
statement */
      if (stmt_num_flag) {  /* print this out only if it's the first line
                               of the SQL statement.  We only want this
                               line to appear once per statement */
        if (verbose)
          fprintf(stderr, "\nTag: %-5.5s    Sequence number: %d\n", \
                g_struct->s_info_ptr->tag,g_struct->s_info_ptr-
>stmt_num);
        fprintf(outstream, "\nTag: %-5.5s    Sequence number: %d\n", \
              g_struct->s_info_ptr->tag,g_struct->s_info_ptr->stmt_num);


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

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

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

      else {                  /* else replace the ; with a \0 and continue */
        *test_semi = '\0';
        strcat (stmt_str.data,input_ln);              /* jen LONG */
        stmt_str.len = strlen( stmt_str.data );       /* jen LONG */
        eostmt = 1;
      }
        fprintf(outstream, "\n%s", input_ln);
        if (verbose)
          fprintf(stderr,"\n%s", input_ln);
      }

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

    }  while (!eostmt);

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

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

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

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

    else                                              /* @d23684 tjg */
      return TPCDBATCH_NONSQL;

  }  /*  Get_SQL_stmt */



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

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

  for (loopvar=0; loopvar<sqlda->sqld; loopvar++)
  {
    switch (sqlda->sqlvar[loopvar].sqltype)
    {
     case SQL_TYP_INTEGER:                        /* INTEGER */
     case SQL_TYP_NINTEGER:
       if ((sqlda->sqlvar[loopvar].sqldata=
              (TPCDBATCH_CHAR *)malloc(sizeof(long))) == NULL)
         mem_error("allocating INTEGER");
       break;
     case SQL_TYP_CHAR:                       /* CHAR */
     case SQL_TYP_NCHAR:
       if ((sqlda->sqlvar[loopvar].sqldata=
              (TPCDBATCH_CHAR *)calloc(256,sizeof(char))) == NULL)
         mem_error("allocating CHAR/VARCHAR");
       break;
     case SQL_TYP_VARCHAR:                        /* VARCHAR */
     case SQL_TYP_NVARCHAR:
```

```
      if ((sqlda->sqlvar[loopvar].sqldata=
            (TPCDBATCH_CHAR *)calloc(4002,sizeof(char))) ==
NULL)
        mem_error("allocating CHAR/VARCHAR");
      break;
    case SQL_TYP_LONG:                    /* LONG VARCHAR */
    case SQL_TYP_NLONG:
      if ((sqlda->sqlvar[loopvar].sqldata=
            (TPCDBATCH_CHAR *)calloc(32702,sizeof(char))) ==
NULL)
        mem_error("allocating VARCHAR/LONG VARCHAR");
      break;
    case SQL_TYP_FLOAT:                   /* FLOAT */
    case SQL_TYP_NFLOAT:
      if ((sqlda->sqlvar[loopvar].sqldata=
            (TPCDBATCH_CHAR *)malloc(sizeof(double))) == NULL)
        mem_error("allocating FLOAT");
      break;
    case SQL_TYP_SMALL:                   /* SMALLINT */
    case SQL_TYP_NSMALL:
      if ((sqlda->sqlvar[loopvar].sqldata=
            (TPCDBATCH_CHAR *)malloc(sizeof(short))) == NULL)
        mem_error("allocating SMALLINT");
      break;
    case SQL_TYP_DECIMAL:                 /* DECIMAL */
    case SQL_TYP_NDECIMAL:
      if ((sqlda->sqlvar[loopvar].sqldata=
            (TPCDBATCH_CHAR *)malloc(20)) == NULL)
        mem_error("allocating DECIMAL");
      break;
    case SQL_TYP_CSTR:                /* VARCHAR (null terminated)
*/
    case SQL_TYP_NCSTR:
      if ((sqlda->sqlvar[loopvar].sqldata=
            (TPCDBATCH_CHAR *)calloc(4001,sizeof(char))) ==
NULL)
        mem_error("allocating CHAR/VARCHAR");
      break;
    case SQL_TYP_DATE:                    /* DATE */
    case SQL_TYP_NDATE:
      if ((sqlda->sqlvar[loopvar].sqldata=
            (TPCDBATCH_CHAR *)calloc(13,sizeof(char))) == NULL)
        mem_error("allocating DATE");
      break;
    case SQL_TYP_TIME:                    /* TIME */
    case SQL_TYP_NTIME:
      if ((sqlda->sqlvar[loopvar].sqldata=
            (TPCDBATCH_CHAR *)calloc(11,sizeof(char))) == NULL)
        mem_error("allocating TIME");
      break;
    case SQL_TYP_STAMP:                   /* TIMESTAMP */
    case SQL_TYP_NSTAMP:
      if ((sqlda->sqlvar[loopvar].sqldata=
            (TPCDBATCH_CHAR *)calloc(29,sizeof(char))) == NULL)
        mem_error("allocating TIMESTAMP");
      break;
   }
   if ((sqlda->sqlvar[loopvar].sqlind=
          (short *)calloc(1,sizeof(short))) == NULL)
     mem_error("allocating indicator");

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


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


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

  int    col_type;              /* Type of column          */

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

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


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

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

        fprintf(outstream, "%*ld  ",col_lengths[col],
              *(long *)(sqlda->sqlvar[col].sqldata));
        break;
       case SQL_TYP_CHAR:
       case SQL_TYP_NCHAR:

        fprintf(outstream, "%-*s  ",col_lengths[col],sqlda-
>sqlvar[col].sqldata);
        break;
       case SQL_TYP_VARCHAR:
       case SQL_TYP_NVARCHAR:
       case SQL_TYP_LONG:
       case SQL_TYP_NLONG:                        /* @d30369 tjg */
        ((struct sqlchar *)sqlda->sqlvar[col].sqldata)->
          data[((struct sqlchar *)sqlda->sqlvar[col].sqldata)->length] = '\0';
        fprintf(outstream, "%-*s  ",
              col_lengths[col],
              ((struct sqlchar *)sqlda->sqlvar[col].sqldata)->data);
        break;
       case SQL_TYP_FLOAT:
       case SQL_TYP_NFLOAT:
       { /* kmw */
        if ( fabs(*(double *)(sqlda->sqlvar[col].sqldata))
                        < TPCDBATCH_PRINT_FLOAT_MAX  )
          fprintf(outstream, "%#*.3f  ",col_lengths[col],
                *(double *)(sqlda->sqlvar[col].sqldata));
        else
          fprintf(outstream, "%*e  ",col_lengths[col],
                *(double *)(sqlda->sqlvar[col].sqldata));
        break;
       }

       case SQL_TYP_SMALL:
       case SQL_TYP_NSMALL:

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

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

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

          temp_ptr = temp_string;

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

          else
            strcpy(decimal_string, " ");

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

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

        break;

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

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

  fprintf(outstream, "\n");

  return;
}


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

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

#if defined (SQLOS2) || defined (SQLWINT)  || defined (SQLWIN) ||
defined (SQLDOS)
  /*@d33143aha*/
  ftime (start_time);
#elif defined(SQLSNI)
  rc = gettimeofday(start_time);
#elif defined (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 (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 next highest 0.1 seconds as
   * required by the TPC-D spec.
   */
  result = (double)(((long)((result + 0.099999) * 10))/10.0);
  return (result);
}
```

```
/*************************************************************
*****************/
/* error_check                                               */
/* This function prints the contents of the sqlca error information          */
/* structure.                                                */
/*************************************************************
*****************/
long error_check(void)
{
  char          buffer[512]="\0";
  unsigned short i;
  struct sqlca   temp_sqlca;       /* temporary sqlca */     /* @d30369 tjg */

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

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

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

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

    EXEC SQL CONNECT ;                                      /* @d28763 tjg */

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

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

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

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

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

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

    }
  }
  return (temp_sqlca.sqlcode);

}  /* error_check */



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

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

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


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

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

  return (string);
}

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

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

```
    *c = (char) tolower( (int) *c );

  return (string);
}


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

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

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


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

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

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

       case 'f':                                /* @d26350 tjg */
       case 'F':
                  strcpy(g_struct->c_l_opt->infile,argv[++loopvar]);
                  break;

       case 'l':                                /* @d26350 tjg */
       case 'L':   strcpy(g_struct->c_l_opt-
>str_file_name,argv[++loopvar]);
                  break;

       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]),"P"))
           g_struct->c_l_opt->update=1;
         if (!strcmp(uppercase(argv[loopvar]),"T1"))
             g_struct->c_l_opt->update=0;
         if (!strcmp(uppercase(argv[loopvar]),"T2"))
             g_struct->c_l_opt->update=2;
         if (!strcmp(uppercase(argv[loopvar]),"T"))
             g_struct->c_l_opt->update=5;
         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;

#ifdef PARALLEL_UPDATES
       case 'i':
         updatePair = atoi (argv[++loopvar]);
         break;

       case 'j':
         function = atoi (argv[++loopvar]);
         break;

       case 'k':
         updateStream = atoi (argv [++loopvar]);
         break;

       case 'x':                                /*DEL jen -x is chunk*/
         deleteChunk = atoi (argv[++loopvar]);       /* to delete for this */
         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);
   }

   if (getenv("TPCD_TMP_DIR") != NULL)
     strcpy(tempdir, getenv("TPCD_TMP_DIR"));
```

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

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

      if (function == 1)
        runUF1_fn (updatePair, updateStream, copyOnOrOff);
      else
        if (function == 2)
        runUF2_fn (updatePair, updateStream, deleteChunk,
copyOnOrOff);
      else {
        fprintf (stderr, "Wrong function to tpcdbatch\n");
        exit (-1);
      }
      exit (0);
    }
  }
#endif /* PARALLEL_UPDATES */

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

  }

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

}



/*************************************************/
/* Converts DECIMAL values to ASCII text         */
/*************************************************/
int sqlrxd2a(                           /*kmw*/
                                        /* C++ */char *decptr,
                                        /* C++ */char *asciiptr,
                                        short prec,
                                        short scal)
{/* */
  int allzero = TRUE;
  /* C++ */char *srcptr;
  unsigned char sign;
  /* C++ */char *targptr, decimal_point = '.';
  int rc = 0;                           /*kmw*/
  int tmpint, src_nibble;
  int count, j, limit[3];

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


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


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

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

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

  }

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



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

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

void init_setup(int argc, char *argv[], struct global_struct *g_struct)
{
  int connect=0;
#ifndef SQLWINT
  char *pid;
#endif
  char temparray[256]="\0";
  int loopvar=0;
  FILE *updateFP;
  FILE *fpSeed;
  char file_name[256] = "\0";
  short seedEntry;
  long  lSeed;
  int i;

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

/*************************************************************
***********/
/* Start the mainline report processing.                     */
/*************************************************************
***********/
  if (!strcmp(g_struct->c_l_opt->infile,"\0")) {
    instream=stdin;
  }
  else {
    instream=NULL;
    if ( (instream = fopen(g_struct->c_l_opt->infile, READMODE)) ==
NULL ) {
      fprintf(outstream, "The input file could not be opened.\n\n");
      fprintf(stdout,"Make sure that the filename is correct.\n");
      fprintf(stdout,"filename = %s\n",g_struct->c_l_opt->infile);
      exit(-1);

    }  /* open the input file if specified  */
  }

  /* determine if we are running from a single node (or a serial setup) */
  /* and therefore want to use semaphores to control the processing of  */
  /* the update functions, or if we are going to be running on multiple */
  /* nodes for the throughput test and want to use the control in the   */
  /* calling script */
  if (getenv("TPCD_RUN_ON_MULTIPLE_NODES") != NULL)
  {
    if
(!strcmp(uppercase(getenv("TPCD_RUN_ON_MULTIPLE_NODES")),"Y
ES"))
    {
      /* okay, we don't want the semaphores */
      g_struct->sem_on = 0;
    }
    else
      g_struct->sem_on = 1;  /* use the semaphores */
  }
  else
  {
    fprintf(stderr,"\nThe TPCD_RUN_ON_MULTIPLE_NODES
environment variable\n");
    fprintf(stderr,"is not set....exiting\n");
    exit(-1);
  }
  /* check to make sure the TPCD_COPY_DIR environment variable is set
*/
  if (getenv("TPCD_COPY_DIR") == NULL)
  {
    fprintf(stderr,"\nThe TPCD_COPY_DIR environment variable is not
set");
    fprintf(stderr,"....exiting\n");
    exit(-1);
  }
  /* IMPORT (begin) - determine whether we should use the IMPORT api
or */
  /* LOAD api for loading into the staging tables */
  if (getenv("TPCD_UPDATE_IMPORT") != NULL)
  {
    if
(!strcmp(uppercase(getenv("TPCD_UPDATE_IMPORT")),"TRUE"))
    {
      iImportStagingTbl = 1;   /* use import */
    }
    /* DJD */
    else if
(!strcmp(uppercase(getenv("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);
    }
    if (getenv("TPCD_AUDIT_DIR") == NULL)
    {
      fprintf(stderr,"\nThe TPCD_AUDIT_DIR 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",
           getenv("TPCD_AUDIT_DIR"),
           getenv("TPCD_PATH_DELIM"),
           getenv("TPCD_PATH_DELIM"));

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

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

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

   /* jenn add code to handle vldb */
#if 0 /*kmw*/
   if ( g_struct->scale_factor >= 1000 )
   {
     sprintf(stmt_str, "SET CURRENT FUCNTION PATH VLDB,
SYSTEM PATH");
     EXEC SQL EXECUTE IMMEDIATE :stmt_str;
     if (sqlca.sqlcode < 0)
     {
       sqlcode = error_check();          /* Don't bother printing the
SQL0100W
                                         error from going through the loop
                                         above one extra time */
     }
   }

#endif /*kmw*/

   /** Get start timestamp for stream **/
   g_struct->stream_start_time = time(NULL);
   strcpy(g_struct->file_time_stamp,
        get_time_stamp(T_STAMP_FORM_2,g_struct-
>stream_start_time));

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

   /* if we are running a throughput test, then we must report the */
   /* stream count information...we will report one file per stream */
   /* and amalgamate them after all streams have completed */
   /* if the number of streams is greater than 0 then this is a throughput
test*/
   if ( g_struct->c_l_opt->intStreamNum > 0 )
   {
     if (g_struct->c_l_opt->update == 2 ||
         g_struct->c_l_opt->update == 5)
     {
       /* update function stream */
       sprintf(file_name,"%s%sstrcntuf.%s",g_struct->run_dir,
             getenv("TPCD_PATH_DELIM"),
             g_struct->file_time_stamp);
     }
     else
     {
       /* query stream */
       sprintf(file_name, "%s%sstrcnt%d.%s",g_struct->run_dir,
             getenv("TPCD_PATH_DELIM"),
             g_struct->c_l_opt->intStreamNum,g_struct->file_time_stamp);
     }
     if( (g_struct->stream_report_file = fopen(file_name, WRITEMODE))
== NULL )
     {
       fprintf(stderr,"\nThe output file for the stream count information\n");
       fprintf(stderr,"could not be opened, make sure the filename is
correct\n");
       fprintf(stderr,"filename = %s\n",file_name);
       exit(-1);
     }
     if (g_struct->c_l_opt->update == 2 ||
         g_struct->c_l_opt->update == 5)
     {
       /* update function stream */
       fprintf(g_struct->stream_report_file,
             "Update function stream starting at %18.18s\n",
```

```
            get_time_stamp(T_STAMP_FORM_1,g_struct-
>stream_start_time));
      }
      else
      {
        /* query stream */
       fprintf(g_struct->stream_report_file,
             "Stream number %d starting at %18.18s\n",
             g_struct->c_l_opt->intStreamNum,
             get_time_stamp(T_STAMP_FORM_1,g_struct-
>stream_start_time));
      }
    }

    /* set up the update_num_file name so that if we do use semaphores, */
    /* we will have a filename to generate the semkey */
    if ( (getenv("TPCD_DBNAME") != NULL) &&
       (getenv("USER") != NULL)       &&
       (getenv("TPCD_AUDIT_DIR") != NULL) )
    {
      sprintf(g_struct->update_num_file, "%s%s%s.%s.update.pair.num",
            getenv("TPCD_AUDIT_DIR"),
            getenv("TPCD_PATH_DELIM"),
            uppercase(getenv("TPCD_DBNAME")),
            lowercase(getenv("USER")));
      sprintf(g_struct->sem_file, "%s.%s.semfile",
            getenv("TPCD_DBNAME"),
            getenv("USER"));
      if (verbose)
      {   /* print out the update pair number file for debugging */
        fprintf(stderr,"\n init_setup: strem %d update pair numb file = %s\n",
            g_struct->c_l_opt->intStreamNum,g_struct->update_num_file);
      }
}
    else
    {
      fprintf(stderr,"\nThe environment is not set up correctly, ensure
that\n");
      fprintf(stderr,"$TPCD_DBNAME, $USER and $TPCD_AUDIT_DIR
are assigned..\n");
      exit(-1);
    }
    /* 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
!= 5 ))
        /* on or onl, but not  */
    {
      updateFP = fopen(g_struct->update_num_file,"r");
      if (updateFP != NULL )
      {
        fscanf(updateFP,"%d",&updatePairStart);
        fclose(updateFP);
        if ( g_struct->c_l_opt->update == 1 )   /* on, 1 update pair */
          updatePairStop = updatePairStart + 1;
        else       /* only, multiple update pairs, stream number will be total */
          updatePairStop = updatePairStart + g_struct->c_l_opt-
>intStreamNum;
        currentUpdatePair = updatePairStart;

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

    return ;

}

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

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

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

    fprintf (outstream,"\n");

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

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

      switch (col_type)
      {
       case SQL_TYP_SMALL:
       case SQL_TYP_NSMALL:                          /* @d30369 tjg */
         col_lengths[col] = TPCDBATCH_MAX (col_name_length,6);
         break;
       case SQL_TYP_INTEGER:
       case SQL_TYP_NINTEGER:
         col_lengths[col] = TPCDBATCH_MAX (col_name_length,11);
         break;
       case SQL_TYP_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 ((m+n), col_name_length);
        /* Special handling for DECIMAL */    /* @d26350 tjg */
        break;

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

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

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

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

}


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

   if (time_pointer == (time_t)NULL)
     temp_stamp = time(NULL);
   else
     temp_stamp = time_pointer;

   tp = localtime(&temp_stamp);
   if (form == T_STAMP_FORM_1)
     strftime(newtime,50,"%x %X",tp);
   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 Ts   = 0;                  /* different TPC-D metrics */
double QppD = 0;
double QthD = 0;
double QphD = 0;

char db_size[8] = "\0";

int scale_factor = 0;

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

/* multiply the scale factor by 1000 to be able to handle scale factors
   less than 1 */
scale_factor = (int)(g_struct->scale_factor * 1000);
switch(scale_factor)
{
 case 10:
   strcpy(db_size,"10MB");
   break;
 case 12:
   strcpy(db_size,"12MB");
   break;
 case 100:
   strcpy(db_size,"100MB");
   break;
 case 1000:
   strcpy(db_size,"1GB");
   break;
 case 3000:
   strcpy(db_size,"3GB");
   break;
 case 8000:
   strcpy (db_size, "8GB");
   break;
 case 10000:
   strcpy(db_size,"10GB");
   break;
 case 30000:
   strcpy(db_size,"30GB");
   break;
 case 100000:
   strcpy(db_size,"100GB");
   break;
 case 1000000:
   strcpy(db_size,"1000GB");
   break;
 default:
   fprintf(stderr,"\nUndefined scale factor\n");
   break;
}


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;
/*   min = s_info_head_ptr;  */
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 %18.18s\n",
          get_time_stamp(T_STAMP_FORM_1,g_struct-
>stream_start_time));
  /* print the stream stop timestamp to the inter file */
  fprintf (outstream, "Stream stop time stamp %18.18s\n",
          get_time_stamp(T_STAMP_FORM_1,g_struct-
>stream_end_time));

  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)
          if (s_info_ptr->elapse_time < (max->elapse_time/1000))
            s_info_ptr->adjusted_time = max->elapse_time/1000;
      }

                                  /* a value was calculated */
      fprintf (outstream,
            "%-5d %-5.5s %15.1f %15.1f%18.18s%18.18s\n",
            s_info_ptr->stmt_num,s_info_ptr->tag,
            s_info_ptr->elapse_time,s_info_ptr->adjusted_time,
            s_info_ptr->start_stamp,s_info_ptr->end_stamp);

      /* 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;
  }


  /* Calculate the arithmetic and geometric means */

  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 = 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 */
  if (g_struct->c_l_opt->update != 2)
  {
    fprintf (outstream, "\nArith. mean %15.3f %15.3f\n",\
          arith_mean,adjusted_a_mean);
    fprintf (outstream, "Geom.  mean %15.3f %15.3f\n",\
          geo_mean,adjusted_g_mean);
```

```
    fprintf (outstream,
          "\n\nMax Qry %-3.3s %15.1f %15.1f%18.18s%18.18s\n",
          max->tag,max->elapse_time,max->adjusted_time,max-
>start_stamp,
          max->end_stamp);
    fprintf (outstream,
          "Min Qry %-3.3s %15.1f %15.1f%18.18s%18.18s\n",
          min->tag,min->elapse_time,min->adjusted_time,min-
>start_stamp,
          min->end_stamp);
  }

  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) */
    Ts = (difftime(g_struct->stream_end_time,g_struct-
>stream_start_time)) + 1;
    QppD = (3600 * g_struct->scale_factor) / adjusted_g_mean;
    QthD = (num_stmt * 3600 * g_struct->scale_factor) / Ts;
    QphD = sqrt(QppD*QthD);

    fprintf (outstream,
          "QppD@%-8.8s = %10.3f\n\nTs %11 = %10.0f\nQthD@%-8.8s
= %10.3f\n\nQphD@%-8.8s = %10.3f\n",
          db_size,QppD, Ts,db_size, QthD,db_size, QphD);
  }

}

/*************************************************************
***/
/* 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 */
}

/* ************ PARALLEL_UPDATES version of runUF1 and runUF2
only */

/************************************************/
/* processing to run the insert update function */
/************************************************/
void runUF1 ( int updatePair, int copyOnOrOff )
{
#ifndef TPCD_NONPARTITIONED
  char statement[3000];
  char sourcedir[256];
#endif

  int split_updates = 2;      /* no. of ways update records are split */
  int concurrent_inserts = 2;  /* jenCI no of concurrent updates to be */
                    /* jenCI run at once*/
  int loop_updates = 1;       /* jenCI no of updates to be run in one  */
                    /* jenCI "concurrent" invocation.  should*/
                    /* jenCI be split_updates / concurrent_inserts*/
  int i, j;
  char myoutstreamfile[256];
  FILE *myoutstream;
  char *buffer;
  char *charptr;

#ifdef SQLWINT
  /* PROCESS_INFORMATION childprocess[100]; */
  char commandline[256];
  HANDLE          su_hSem;
  char            UF1_semfile[256];
#else
  int childpid[100];
  int             su_semid; /* semaphore for controlling split updates*/
  key_t           su_semkey; /* key to generate semid */
#endif

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

#ifndef TPCD_NONPARTITIONED
      /* for PE and not table function we call the load_update script to
       * perform the appropriate load across the nodes. For SE, the load
       * and insert are all done by the runUF1_fn function.
       */
  if  ((!(charptr = getenv("TPCD_UPDATE_IMPORT")))   ||
       ((strncmp(charptr,"TF", 2))          &&
        (strncmp(charptr,"tf", 2))
       )
      )
    {
      /* determine the directory in which to find the script   */
      if (getenv("TPCD_PELOAD_DIR") != NULL)
        strcpy(sourcedir,getenv("TPCD_PELOAD_DIR"));
      else
        strcpy(sourcedir,".");

      sprintf(statement, "%s/load_update %d 1",sourcedir,updatePair);
      if (system(statement))
      {
        fprintf (outstream,
              "load_update failed for UF1, examine UF1.log for cause.
Exiting.\n");
        if (verbose)
          fprintf (stderr,
              "load_update failed for UF1, examine UF1.log for cause.
Exiting.\n");
        exit (-1);
      }
      fprintf (outstream, "load_update finished for UF1.\n");
    }
#endif  /* not TPCD_NONPARTITIONED */

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

#ifndef SQLWINT
  /*    we will use the first flat file to generate the semaphore key */
  if (getenv("TPCD_FLATFILES") != NULL)
  {
#ifndef TPCD_NONPARTITIONED
    /* this is assuming that you will be running this from 0th node */
    sprintf(sourcefile, "%s%corder.tbl.u%d.00000.new",
          getenv("TPCD_FLATFILES"), PATH_DELIM,updatePair);
#else
    sprintf(sourcefile, "%s%corder.tbl.u%d.0",
```

```
            getenv("TPCD_FLATFILES"), PATH_DELIM, updatePair);
#endif
  }
  else
  {
    fprintf (stderr, "TPCD_FLATFILES is not defined.\n");
    exit (-1);
  }

  su_semkey = ftok (sourcefile, 'J');
  if ( (su_semid = semget (su_semkey, 1,
IPC_CREAT|S_IRUSR|S_IWUSR)) < 0)
  {
    fprintf (stderr, "Can't get semaphore! semget failed: errno = %d\n",
            errno);
    exit (-1);
  }
#else  /* SQLWINT */
  sprintf (UF1_semfile, "%s.%s.UF1.semfile",
          getenv("TPCD_DBNAME"), getenv("USER"));
  fprintf(stderr,"UF1 semfile = %s\n",UF1_semfile);
  su_hSem = CreateSemaphore(NULL, 0,
                        concurrent_inserts,                 /*jenCI*/
                        (LPCTSTR)(UF1_semfile));
  if (su_hSem == NULL)
  {
    fprintf(stderr,
          "CreateSemaphore (ready semaphore) failed, GetLastError: %d,
quitting\n",
          GetLastError());
    exit(-1);
  }
#endif /* SQLWINT */
  if (verbose) fprintf(stderr,"Semaphore created successfully!\n");

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

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

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

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

  if( (outstream = fopen(outstreamfilename, APPENDMODE)) == NULL )
  {
    fprintf(stderr,"\nThe output file could not be opened.  ");
    fprintf(stderr,"Make sure that the filename is correct.\n");
    fprintf(stderr,"filename = %s\n",outstreamfilename);
    exit(-1);
  }
  /* fixing multiple header problem
    wlc 081397 */

  /* combine the output files from the different streams */
  if ((buffer = malloc (BUFSIZE)) == NULL)
  {
    exit (-1);
  }
  for (i=0; i < concurrent_inserts; i++)                    /*jenCI*/
  {
    sprintf (myoutstreamfile, UF1OUTSTREAMPATTERN, tempdir,
PATH_DELIM,
            updatePair, i);
    if ( (myoutstream = fopen (myoutstreamfile, READMODE)) ==
NULL) {
      fprintf (stderr,
            "\nRunUF1: The output file '%s' for update pair %d set %d
could not be opened. runUF1\n",
            myoutstreamfile,updatePair,i);
      fprintf (outstream,
            "\nRunUF1: The output file '%s' for update pair %d set %d
could not be opened. runUF1\n",
            myoutstreamfile,updatePair,i);
      exit (-1);
    }
    /* copy the while output file for stream i into the real output file */
    do
    {
      j=fread (buffer, sizeof (char), BUFSIZE, myoutstream);
      fwrite (buffer, sizeof (char), j, outstream);
    } while (!feof (myoutstream));
    fclose (myoutstream);
  }

  fprintf( outstream,"UF1 for update pair %d complete\n",updatePair);
}
void runUF1_fn ( int updatePair, int i, int copyOnOrOff )
{
  int rc = 0;
  int UF1style = 0;
```

```
  int num_physical;
  int num_ln_per_pn;
  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 */
EXEC SQL BEGIN DECLARE SECTION;
  long chunk = 0;              /* jenCI counter for within the set of chunks*/
EXEC SQL END DECLARE SECTION;

  long sqlcode;
  int maxwait;

#ifndef SQLWINT
  int               su_semid;              /* semaphore for controlling split
updates*/
  key_t             su_semkey;             /* key to generate semid */
#else
  HANDLE            su_hSem;
  char              UF1_semfile[256];
#endif

  char myoutstreamfile[256];
  FILE *myoutstream;

  /* Get ready to start logging diagnostic output */
  sprintf (myoutstreamfile, UF1OUTSTREAMPATTERN, tempdir,
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
%18.18s\n",
          updatePair, i,
get_time_stamp(T_STAMP_FORM_1,(time_t)NULL));

  /* must reread the environment variable since they are not inherited by */
  /* children */
  if (getenv("TPCD_UF1_STYLE") != NULL)
  {
    if (!strcmp(uppercase(getenv("TPCD_UF1_STYLE")),"LOAD"))
    {
      UF1style = 1;   /* use staged load */
      fprintf(myoutstream, "Using staged LOAD to perform UF1\n");
    }
    if (!strcmp(uppercase(getenv("TPCD_UF1_STYLE")),"IMPORT"))
    {
      UF1style = 2;   /* use staged import */
      fprintf(myoutstream, "Using staged IMPORT to perform UF1\n");
    }
    if (!strcmp(uppercase(getenv("TPCD_UF1_STYLE")),"TF"))
    {
      UF1style = 3;   /* use direct table functions */
      fprintf(myoutstream, "Using Table functions to perform UF1\n");
      fprintf(myoutstream, "Not supported right now!\n");
    }
    if
(!strcmp(uppercase(getenv("TPCD_UF1_STYLE")),"DIRECTIMPORT"))
    {
      UF1style = 4;   /* use direct import */
      fprintf(myoutstream, "Using direct IMPORT to perform UF1\n");
      fprintf(myoutstream, "Not supported right now!\n");
    }
  }
  else
  {
    UF1style = 1;   /* assume staged load by defaul */
    fprintf(myoutstream, "Using staged LOAD to perform UF1\n");
  }
  if (UF1style <= 0)
  {
      fprintf(myoutstream, "Pick a UF1 style please!\n");
  }

  /* If running NONPARTITIONED and using a staging mechanism, need
to stage the tables */
  if (getenv ("TPCD_PHYS_NODE") != NULL)
    num_physical = atoi (getenv ("TPCD_PHYS_NODE"));
  if (getenv ("TPCD_LN_PER_PN") != NULL)
    num_ln_per_pn = atoi (getenv ("TPCD_LN_PER_PN"));
  if (num_physical * num_ln_per_pn == 1 && UF1style < 3)
  {
    /* need to do the staging inline - call a helper function */
    /*      rc = runUF1_buildstagingtables ( updatePair, i, copyOnOrOff,
UF1_style) */
    if ( rc != 0)
    {
      fprintf(myoutstream, "Populating staging tables failed.\n");
      goto UF1_exit;
    }
  }

  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 :dbname;
  else
    EXEC SQL CONNECT TO :dbname USER :userid USING :passwd;
  error_check();
  if (sqlca.sqlcode < 0)
  {
    rc=-1;
    goto UF1_exit;
  }

  /* Start processing each chunk in my range */
  for ( chunk = startChunk; chunk <= stopChunk; chunk++ )
/*jenCI*/
  {                                                          /*jenCI*/
    /* wlc  062797 */
    sqlcode = SQL_RC_E911;
    maxwait = 1;
```

```
    /* Loop to handle any deadlocks */
    while (sqlcode == SQL_RC_E911 && maxwait <= MAXWAIT &&
rc==0)
    {
      sqlcode = 0;

     EXEC SQL INSERT INTO TPCD.ORDERS SELECT
O_ORDERKEY,O_CUSTKEY,O_ORDERSTATUS,O_TOTALPRICE,O_
ORDERDATE,

O_ORDERPRIORITY,O_CLERK,O_SHIPPRIORITY,O_COMMENT
FROM TPCDTEMP.ORDERS_NEW WHERE
       APP_ID = :chunk;

      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",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,chunk,sqlcode);
        rc = -1;
      }
      else
      {
        /* Everything worked with ORDERS, proceed with LINEITEM */
        /* report the number of row inserted */
        fprintf(myoutstream, "%ld rows inserted into TPCD.ORDERS at
%18.18s\n",
              sqlca.sqlerrd[2],
              get_time_stamp(T_STAMP_FORM_1,(time_t)NULL));

        sqlcode = 0;

        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 =
:chunk;

        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, chunk);
          SleepSome(UF_DEADLOCK_SLEEP);
          maxwait++;                        /* jen DEADLOCK */
        }
        else if (sqlcode < 0)
        {
          fprintf (myoutstream,
                "\nAn error occurred inserting into TPCD.LINEITEM\n");
          fprintf (myoutstream,
                "for update pair number %d chunk %d ...Exiting\n",
                updatePair, chunk);
        }
        else
        {
#ifdef UF2DEBUG
          fprintf (myoutstream, "lineitem insert succeeded\n");
          fflush(myoutstream);
#endif
          /* report the number of row deleted */
          fprintf(myoutstream, "%ld rows inserted into TPCD.LINEITEM
at %18.18s\n",
                sqlca.sqlerrd[2],
                get_time_stamp(T_STAMP_FORM_1,(time_t)NULL));

          fprintf( myoutstream,
                "UF1 for update pair %d chunk %d complete at
%18.18s\n\n",
                updatePair, chunk,
get_time_stamp(T_STAMP_FORM_1,(time_t)NULL));

          rc=0;
          EXEC SQL COMMIT WORK;
          error_check();
        }
      } /* process lineitem INSERTs */
    } /* while loop for deadlocks */
  } /* while processing chunks */

  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 first flat file to generate the semaphore key */
  if (getenv("TPCD_FLATFILES") != NULL) {
#ifndef TPCD_NONPARTITIONED
    /* this is assuming that you will be running this from 0th node */
    sprintf(sourcefile, "%s%corder.tbl.u%d.00000.new",
          getenv("TPCD_FLATFILES"), PATH_DELIM,updatePair);
#else
    sprintf(sourcefile, "%s%corder.tbl.u%d.0",
          getenv("TPCD_FLATFILES"), PATH_DELIM,updatePair);
#endif
  }
  else
  {
    fprintf (stderr, "TPCD_FLATFILES is not defined.\n");
    exit (-1);
  }

  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 */
}

/************************************************/
/* processing to run the delete update function */
/************************************************/
void runUF2 ( int updatePair, int copyOnOrOff )
{
#ifndef TPCD_NONPARTITIONED
  char statement[3000];
  char sourcedir[256];
#endif
  int split_deletes = 1;    /*   no. of ways update records are split
@dxxxxxhar */
  int numChunks = 1;    /* number of database partitions DELjen */
  int i, j, c;
  char myoutstreamfile[256];
  FILE *myoutstream;
/* char *buffer;   *kmw* */
#ifdef SQLWINT
  char commandline[256];
  HANDLE              su_hSem;
  char               UF2_semfile[256];
#else
  int childpid[100];
  int                su_semid; /* semaphore for controlling split updates*/
  key_t               su_semkey; /* key to generate semid */
#endif

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

  /* v2 : just read the variable TPCD_SPLIT_DELETES */
  if (getenv ("TPCD_SPLIT_DELETES") != NULL)
    split_deletes = atoi (getenv ("TPCD_SPLIT_DELETES"));
#ifndef TPCD_NONPARTITIONED
  /* DELjen in mln/mpp, the TPCD_SPLIT_DELETES really is the */
  /* number of chunks per data partitions whereas split_deletes in the
code*/
  /* later on is used to mean the number of partitons....so set numChunks
*/
  /* to be TPCD_SPLIT_DELETES and split_deletes = number of
partitions */
  /* mln or mpp: deletes are done by calling tpcdbatch on each logical
node */
  if ((getenv ("TPCD_PHYS_NODE") != NULL) &&
     (getenv ("TPCD_LN_PER_PN") != NULL))
  {
    numChunks = split_deletes;
    split_deletes = atoi (getenv ("TPCD_PHYS_NODE")) *
      atoi (getenv ("TPCD_LN_PER_PN"));
  }
#endif

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

#ifdef TPCD_NONPARTITIONED
#ifndef SQLWINT
  /*    we will use the first flat file to generate the semaphore key */
  if (getenv("TPCD_FLATFILES") != NULL)
    sprintf(sourcefile, "%s%cdelete.%d.0",
          getenv("TPCD_FLATFILES"), PATH_DELIM, updatePair);
  else
    sprintf(sourcefile,".%cdelete.%d.0", PATH_DELIM,
          updatePair);

  su_semkey = ftok (sourcefile, 'J');
  if ( (su_semid = semget (su_semkey, 1,
IPC_CREAT|S_IRUSR|S_IWUSR)) < 0)
  {
    fprintf (stderr, "UF2 Can't get semaphore! semget failed: errno =
%d\n",
          errno);
    exit (-1);
  }
#else
  sprintf (UF2_semfile, "%s.%s.UF2.semfile",
        getenv("TPCD_DBNAME"), getenv("USER"));
  fprintf(stderr,"UF2 semfile = %s\n",UF2_semfile);
  su_hSem = CreateSemaphore(NULL, 0,
                  split_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 < split_deletes; i++)
  {
#ifndef SQLWINT
    if ((childpid[i] = fork()) == 0)
    {
```

```
        runUF2_fn (updatePair, i, 0, copyOnOrOff);   /* DELjen use
deleteChunk */
        /* parm = 0 when running smp/uni only used for mln/mpp since this
*/
        /* call handles the chunking in smp/uni */
      }
      else
      {
        /* This is the parent */
        if (verbose)
          fprintf (stderr, "stream #%d started with pid %d\n", i, childpid[i]);
      }
#else
      {
        /*  SECURITY_ATTRIBUTES sec_process;
            SECURITY_ATTRIBUTES sec_thread; */

#if 0
        sprintf (commandline,
              "\\tpcdaudit\\auditruns\\tpcdbatch -z -d %s -i %d -j 2 -k %d",
              dbname, updatePair, i );
        if (CreateProcess("tpcdbatch", commandline, NULL, NULL, FALSE,
                    DETACHED_PROCESS, NULL, NULL, NULL,
&childprocess[i])
            /*
              if (CreateProcess(NULL, commandline, NULL, NULL, FALSE,
              0, NULL, NULL, NULL, &childprocess[i]) */
            == FALSE)

        {
          fprintf (stderr, "CreateProcess failed: GetLastError: %d\n",
                GetLastError());
          exit (-1);
        }
        else
        {
          if (verbose)
            fprintf (stderr, "Stream #%d started with pid %d\n",
                  childprocess[i].dwProcessId);
        }
#endif
        sprintf (commandline,
            "start /b %s\\auditruns\\tpcdbatch.exe -z -d %s -i %d -j 2 -k %d -x
0",
            getenv("TPCD_AUDIT_DIR"), dbname, updatePair, i ); /* aph */
        /* the -x parm should be passed at 0...not 100% sure of this jen */
        system (commandline);
        sleep (UF2_SLEEP);
      }
#endif
    }

    /* All children have been created, now wait for them to finish */
#ifndef SQLWINT
    fprintf(stderr, "About to wait on the semaphore...\n");
    if (sem_op (su_semid, 0, split_deletes * -1) != 0)              /*jenSEM*/
    {                                                       /*jenSEM*/
      fprintf(stderr,
            "Failure to update wait on delete semaphone with %d children\n",
            split_deletes);
      exit(1);
    }                                                       /*jenSEM*/
    semctl (su_semid, 0, IPC_RMID, 0);
#else
    for (i = 0; i < split_deletes; i++)
    {
      if (verbose)
      {
        fprintf(stderr,"About to wait again ...Sets to wait for %d\n",
              split_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

#else  /* non TPCD_NONPARTITIONED */
    if (getenv("TPCD_PELOAD_DIR") != NULL)
      strcpy(sourcedir,getenv("TPCD_PELOAD_DIR"));
    else
      strcpy(sourcedir,".");

    sprintf (statement, "%s/load_update %d 2 %d", sourcedir, updatePair,
          inlistmax);
    if (system(statement))
    {
      fprintf (stderr, "load_update failed for UF2, examine UF2.log for
cause. Exiting.\n");
      exit (-1);
    }
    fprintf (outstream, "load_update finished for UF2.\n");

#endif /* TPCD_NONPARTITIONED */

    if( (outstream = fopen(outstreamfilename, APPENDMODE)) == NULL )
{
      fprintf(stderr,"\nRunUF2 The output file could not be opened.  ");
      fprintf(stderr,"Make sure that the filename is correct.\n");
      fprintf(stderr,"filename = %s\n",outstreamfilename);
      exit(-1);
    }

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

void runUF2_fn ( int updatePair, int i, int deleteChunk, int copyOnOrOff )
{
    int rc = 0;
    int j;

    long sqlcode;
    int maxwait;

#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;
    long key;
    char keystr[32];
    int done = FALSE;

    EXEC SQL BEGIN DECLARE SECTION;
      long inl001,inl002,inl003,inl004,inl005,inl006,inl007,inl008,inl009;
      long
inl010,inl011,inl012,inl013,inl014,inl015,inl016,inl017,inl018,inl019;
      long
inl020,inl021,inl022,inl023,inl024,inl025,inl026,inl027,inl028,inl029;
```

```
        long
inl030,inl031,inl032,inl033,inl034,inl035,inl036,inl037,inl038,inl039;
        long
inl040,inl041,inl042,inl043,inl044,inl045,inl046,inl047,inl048,inl049;
        long
inl050,inl051,inl052,inl053,inl054,inl055,inl056,inl057,inl058,inl059;
        long
inl060,inl061,inl062,inl063,inl064,inl065,inl066,inl067,inl068,inl069;
        long
inl070,inl071,inl072,inl073,inl074,inl075,inl076,inl077,inl078,inl079;
        long
inl080,inl081,inl082,inl083,inl084,inl085,inl086,inl087,inl088,inl089;
        long
inl090,inl091,inl092,inl093,inl094,inl095,inl096,inl097,inl098,inl099;
        long
inl100,inl101,inl102,inl103,inl104,inl105,inl106,inl107,inl108,inl109;
        long
inl110,inl111,inl112,inl113,inl114,inl115,inl116,inl117,inl118,inl119;
        long
inl120,inl121,inl122,inl123,inl124,inl125,inl126,inl127,inl128,inl129;
        long
inl130,inl131,inl132,inl133,inl134,inl135,inl136,inl137,inl138,inl139;
        long
inl140,inl141,inl142,inl143,inl144,inl145,inl146,inl147,inl148,inl149;
        long
inl150,inl151,inl152,inl153,inl154,inl155,inl156,inl157,inl158,inl159;
        long
inl160,inl161,inl162,inl163,inl164,inl165,inl166,inl167,inl168,inl169;
        long
inl170,inl171,inl172,inl173,inl174,inl175,inl176,inl177,inl178,inl179;
        long
inl180,inl181,inl182,inl183,inl184,inl185,inl186,inl187,inl188,inl189;
        long
inl190,inl191,inl192,inl193,inl194,inl195,inl196,inl197,inl198,inl199;
        long
inl200,inl201,inl202,inl203,inl204,inl205,inl206,inl207,inl208,inl209;
        long
inl210,inl211,inl212,inl213,inl214,inl215,inl216,inl217,inl218,inl219;
        long
inl220,inl221,inl222,inl223,inl224,inl225,inl226,inl227,inl228,inl229;
        long
inl230,inl231,inl232,inl233,inl234,inl235,inl236,inl237,inl238,inl239;
        long
inl240,inl241,inl242,inl243,inl244,inl245,inl246,inl247,inl248,inl249;
        long
inl250,inl251,inl252,inl253,inl254,inl255,inl256,inl257,inl258,inl259;
        long
inl260,inl261,inl262,inl263,inl264,inl265,inl266,inl267,inl268,inl269;
        long
inl270,inl271,inl272,inl273,inl274,inl275,inl276,inl277,inl278,inl279;
        long
inl280,inl281,inl282,inl283,inl284,inl285,inl286,inl287,inl288,inl289;
        long
inl290,inl291,inl292,inl293,inl294,inl295,inl296,inl297,inl298,inl299;
        long
inl300,inl301,inl302,inl303,inl304,inl305,inl306,inl307,inl308,inl309;
        long
inl310,inl311,inl312,inl313,inl314,inl315,inl316,inl317,inl318,inl319;
        long
inl320,inl321,inl322,inl323,inl324,inl325,inl326,inl327,inl328,inl329;
        long
inl330,inl331,inl332,inl333,inl334,inl335,inl336,inl337,inl338,inl339;
        long
inl340,inl341,inl342,inl343,inl344,inl345,inl346,inl347,inl348,inl349;
        long
inl350,inl351,inl352,inl353,inl354,inl355,inl356,inl357,inl358,inl359;
        long
inl360,inl361,inl362,inl363,inl364,inl365,inl366,inl367,inl368,inl369;
        long
inl370,inl371,inl372,inl373,inl374,inl375,inl376,inl377,inl378,inl379;
        long
inl380,inl381,inl382,inl383,inl384,inl385,inl386,inl387,inl388,inl389;
        long
inl390,inl391,inl392,inl393,inl394,inl395,inl396,inl397,inl398,inl399;
        long
inl400,inl401,inl402,inl403,inl404,inl405,inl406,inl407,inl408,inl409;
        long
inl410,inl411,inl412,inl413,inl414,inl415,inl416,inl417,inl418,inl419;
        long
inl420,inl421,inl422,inl423,inl424,inl425,inl426,inl427,inl428,inl429;
        long
inl430,inl431,inl432,inl433,inl434,inl435,inl436,inl437,inl438,inl439;
        long
inl440,inl441,inl442,inl443,inl444,inl445,inl446,inl447,inl448,inl449;
        long
inl450,inl451,inl452,inl453,inl454,inl455,inl456,inl457,inl458,inl459;
        long
inl460,inl461,inl462,inl463,inl464,inl465,inl466,inl467,inl468,inl469;
        long
inl470,inl471,inl472,inl473,inl474,inl475,inl476,inl477,inl478,inl479;
        long
inl480,inl481,inl482,inl483,inl484,inl485,inl486,inl487,inl488,inl489;
        long
inl490,inl491,inl492,inl493,inl494,inl495,inl496,inl497,inl498,inl499;
        long
inl500,inl501,inl502,inl503,inl504,inl505,inl506,inl507,inl508,inl509;
        long
inl510,inl511,inl512,inl513,inl514,inl515,inl516,inl517,inl518,inl519;
        long
inl520,inl521,inl522,inl523,inl524,inl525,inl526,inl527,inl528,inl529;
        long
inl530,inl531,inl532,inl533,inl534,inl535,inl536,inl537,inl538,inl539;
        long
inl540,inl541,inl542,inl543,inl544,inl545,inl546,inl547,inl548,inl549;
        long
inl550,inl551,inl552,inl553,inl554,inl555,inl556,inl557,inl558,inl559;
        long
inl560,inl561,inl562,inl563,inl564,inl565,inl566,inl567,inl568,inl569;
        long
inl570,inl571,inl572,inl573,inl574,inl575,inl576,inl577,inl578,inl579;
        long
inl580,inl581,inl582,inl583,inl584,inl585,inl586,inl587,inl588,inl589;
        long
inl590,inl591,inl592,inl593,inl594,inl595,inl596,inl597,inl598,inl599;
        long
inl600,inl601,inl602,inl603,inl604,inl605,inl606,inl607,inl608,inl609;
        long
inl610,inl611,inl612,inl613,inl614,inl615,inl616,inl617,inl618,inl619;
        long
inl620,inl621,inl622,inl623,inl624,inl625,inl626,inl627,inl628,inl629;
        long
inl630,inl631,inl632,inl633,inl634,inl635,inl636,inl637,inl638,inl639;
        long
inl640,inl641,inl642,inl643,inl644,inl645,inl646,inl647,inl648,inl649;
        long
inl650,inl651,inl652,inl653,inl654,inl655,inl656,inl657,inl658,inl659;
        long
inl660,inl661,inl662,inl663,inl664,inl665,inl666,inl667,inl668,inl669;
        long
inl670,inl671,inl672,inl673,inl674,inl675,inl676,inl677,inl678,inl679;
        long
inl680,inl681,inl682,inl683,inl684,inl685,inl686,inl687,inl688,inl689;
        long
inl690,inl691,inl692,inl693,inl694,inl695,inl696,inl697,inl698,inl699;
        long
inl700,inl701,inl702,inl703,inl704,inl705,inl706,inl707,inl708,inl709;
  EXEC SQL END DECLARE SECTION;

#ifdef TPCD_NONPARTITIONED
  sprintf (myoutstreamfile, UF2OUTSTREAMPATTERN, tempdir,
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 runUF2_fn.\n",
            myoutstreamfile,updatePair,i);
    rc=-1;
    goto UF2_exit;
  }

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

  fprintf( myoutstream,
          "\nUF2 for update pair %d set %d starting at %18.18s\n",
          updatePair, i,
get_time_stamp(T_STAMP_FORM_1,(time_t)NULL));
#else
  /*DELjen not TPCD_NONPARTITIONED use different outstream name
*/
  sprintf (myoutstreamfile, UF2OUTSTREAMPATTERN, tempdir,
PATH_DELIM,
          updatePair, i,deleteChunk);
  if ( (myoutstream = fopen (myoutstreamfile, WRITEMODE)) == NULL)
  {
    fprintf (stderr,
            "\nThe output file '%s' for update pair %d node %d chunk %d
could not be opened in runUF2_fn.\n",
            myoutstreamfile,updatePair,i,deleteChunk);
    rc=-1;
    goto UF2_exit;
  }

  fprintf( myoutstream,
          "\nUF2 for update pair %d node %d chunk %d starting at
%18.18s\n",

updatePair,i,deleteChunk,get_time_stamp(T_STAMP_FORM_1,(time_t)N
ULL));
  /* endDELjen */
#endif

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

  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,"\nStart the DB2 Database Manager first\n");
    exit (-1);
  }
  error_check();

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

#ifdef TPCD_NONPARTITIONED
  if (getenv("TPCD_FLATFILES") != NULL)
    sprintf(sourcefile, "%s%cdelete.%d.%d",
          getenv("TPCD_FLATFILES"),
          PATH_DELIM, updatePair, i);
#else
  sprintf(sourcefile,"%s%cdelete.%d.%05d.%d",     /*DELjen added
deleteChunk */
        getenv("TPCD_FLATFILES"),
        PATH_DELIM, updatePair, i,deleteChunk);
#endif

  if ( (src_fh = fopen (sourcefile, READMODE)) == NULL )
  {
#ifdef TPCD_NONPARTITIONED
    fprintf (myoutstream, "\nThe source file '%s' for update pair %d set %d
could not be opened.\n",
          sourcefile, updatePair, i);
#else
    fprintf (myoutstream,
          "\nThe source file '%s' for update pair %d node %d chunk %d
could not be opened.\n",
          sourcefile, updatePair, i, deleteChunk);
#endif
    rc = -1;
    goto UF2_conn_reset;
  }

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

  /* can't have more inlist items than host variables */
  if (inlistmax > 709)
  {
    inlistmax = 709;  /*  hardcoded since it is static */
  }

  while ( ! done && rc==0 )
  {
    exec sql begin declare section;
    short
ind001,ind002,ind003,ind004,ind005,ind006,ind007,ind008,ind009=-1;
    short
ind010,ind011,ind012,ind013,ind014,ind015,ind016,ind017,ind018,ind01
9=-1;
    short
ind020,ind021,ind022,ind023,ind024,ind025,ind026,ind027,ind028,ind02
9=-1;
    short
ind030,ind031,ind032,ind033,ind034,ind035,ind036,ind037,ind038,ind03
9=-1;
    short
ind040,ind041,ind042,ind043,ind044,ind045,ind046,ind047,ind048,ind04
9=-1;
    short
ind050,ind051,ind052,ind053,ind054,ind055,ind056,ind057,ind058,ind05
9=-1;
    short
ind060,ind061,ind062,ind063,ind064,ind065,ind066,ind067,ind068,ind06
9=-1;
    short
ind070,ind071,ind072,ind073,ind074,ind075,ind076,ind077,ind078,ind07
9=-1;
    short
ind080,ind081,ind082,ind083,ind084,ind085,ind086,ind087,ind088,ind08
9=-1;
    short
ind090,ind091,ind092,ind093,ind094,ind095,ind096,ind097,ind098,ind09
9=-1;
    short
ind100,ind101,ind102,ind103,ind104,ind105,ind106,ind107,ind108,ind10
9=-1;
    short
ind110,ind111,ind112,ind113,ind114,ind115,ind116,ind117,ind118,ind11
9=-1;
    short
ind120,ind121,ind122,ind123,ind124,ind125,ind126,ind127,ind128,ind12
9=-1;
```

```
    short
ind130,ind131,ind132,ind133,ind134,ind135,ind136,ind137,ind138,ind13
9=-1;
    short
ind140,ind141,ind142,ind143,ind144,ind145,ind146,ind147,ind148,ind14
9=-1;
    short
ind150,ind151,ind152,ind153,ind154,ind155,ind156,ind157,ind158,ind15
9=-1;
    short
ind160,ind161,ind162,ind163,ind164,ind165,ind166,ind167,ind168,ind16
9=-1;
    short
ind170,ind171,ind172,ind173,ind174,ind175,ind176,ind177,ind178,ind17
9=-1;
    short
ind180,ind181,ind182,ind183,ind184,ind185,ind186,ind187,ind188,ind18
9=-1;
    short
ind190,ind191,ind192,ind193,ind194,ind195,ind196,ind197,ind198,ind19
9=-1;
    short
ind200,ind201,ind202,ind203,ind204,ind205,ind206,ind207,ind208,ind20
9=-1;
    short
ind210,ind211,ind212,ind213,ind214,ind215,ind216,ind217,ind218,ind21
9=-1;
    short
ind220,ind221,ind222,ind223,ind224,ind225,ind226,ind227,ind228,ind22
9=-1;
    short
ind230,ind231,ind232,ind233,ind234,ind235,ind236,ind237,ind238,ind23
9=-1;
    short
ind240,ind241,ind242,ind243,ind244,ind245,ind246,ind247,ind248,ind24
9=-1;
    short
ind250,ind251,ind252,ind253,ind254,ind255,ind256,ind257,ind258,ind25
9=-1;
    short
ind260,ind261,ind262,ind263,ind264,ind265,ind266,ind267,ind268,ind26
9=-1;
    short
ind270,ind271,ind272,ind273,ind274,ind275,ind276,ind277,ind278,ind27
9=-1;
    short
ind280,ind281,ind282,ind283,ind284,ind285,ind286,ind287,ind288,ind28
9=-1;
    short
ind290,ind291,ind292,ind293,ind294,ind295,ind296,ind297,ind298,ind29
9=-1;
    short
ind300,ind301,ind302,ind303,ind304,ind305,ind306,ind307,ind308,ind30
9=-1;
    short
ind310,ind311,ind312,ind313,ind314,ind315,ind316,ind317,ind318,ind31
9=-1;
    short
ind320,ind321,ind322,ind323,ind324,ind325,ind326,ind327,ind328,ind32
9=-1;
    short
ind330,ind331,ind332,ind333,ind334,ind335,ind336,ind337,ind338,ind33
9=-1;
    short
ind340,ind341,ind342,ind343,ind344,ind345,ind346,ind347,ind348,ind34
9=-1;
    short
ind350,ind351,ind352,ind353,ind354,ind355,ind356,ind357,ind358,ind35
9=-1;
    short
ind360,ind361,ind362,ind363,ind364,ind365,ind366,ind367,ind368,ind36
9=-1;
    short
ind370,ind371,ind372,ind373,ind374,ind375,ind376,ind377,ind378,ind37
9=-1;
    short
ind380,ind381,ind382,ind383,ind384,ind385,ind386,ind387,ind388,ind38
9=-1;
    short
ind390,ind391,ind392,ind393,ind394,ind395,ind396,ind397,ind398,ind39
9=-1;
    short
ind400,ind401,ind402,ind403,ind404,ind405,ind406,ind407,ind408,ind40
9=-1;
    short
ind410,ind411,ind412,ind413,ind414,ind415,ind416,ind417,ind418,ind41
9=-1;
    short
ind420,ind421,ind422,ind423,ind424,ind425,ind426,ind427,ind428,ind42
9=-1;
    short
ind430,ind431,ind432,ind433,ind434,ind435,ind436,ind437,ind438,ind43
9=-1;
    short
ind440,ind441,ind442,ind443,ind444,ind445,ind446,ind447,ind448,ind44
9=-1;
    short
ind450,ind451,ind452,ind453,ind454,ind455,ind456,ind457,ind458,ind45
9=-1;
    short
ind460,ind461,ind462,ind463,ind464,ind465,ind466,ind467,ind468,ind46
9=-1;
    short
ind470,ind471,ind472,ind473,ind474,ind475,ind476,ind477,ind478,ind47
9=-1;
    short
ind480,ind481,ind482,ind483,ind484,ind485,ind486,ind487,ind488,ind48
9=-1;
    short
ind490,ind491,ind492,ind493,ind494,ind495,ind496,ind497,ind498,ind49
9=-1;
    short
ind500,ind501,ind502,ind503,ind504,ind505,ind506,ind507,ind508,ind50
9=-1;
    short
ind510,ind511,ind512,ind513,ind514,ind515,ind516,ind517,ind518,ind51
9=-1;
    short
ind520,ind521,ind522,ind523,ind524,ind525,ind526,ind527,ind528,ind52
9=-1;
    short
ind530,ind531,ind532,ind533,ind534,ind535,ind536,ind537,ind538,ind53
9=-1;
    short
ind540,ind541,ind542,ind543,ind544,ind545,ind546,ind547,ind548,ind54
9=-1;
    short
ind550,ind551,ind552,ind553,ind554,ind555,ind556,ind557,ind558,ind55
9=-1;
    short
ind560,ind561,ind562,ind563,ind564,ind565,ind566,ind567,ind568,ind56
9=-1;
    short
ind570,ind571,ind572,ind573,ind574,ind575,ind576,ind577,ind578,ind57
9=-1;
    short
ind580,ind581,ind582,ind583,ind584,ind585,ind586,ind587,ind588,ind58
9=-1;
    short
ind590,ind591,ind592,ind593,ind594,ind595,ind596,ind597,ind598,ind59
9=-1;
    short
ind600,ind601,ind602,ind603,ind604,ind605,ind606,ind607,ind608,ind60
9=-1;
```

```
    short
ind610,ind611,ind612,ind613,ind614,ind615,ind616,ind617,ind618,ind61
9=-1;
    short
ind620,ind621,ind622,ind623,ind624,ind625,ind626,ind627,ind628,ind62
9=-1;
    short
ind630,ind631,ind632,ind633,ind634,ind635,ind636,ind637,ind638,ind63
9=-1;
    short
ind640,ind641,ind642,ind643,ind644,ind645,ind646,ind647,ind648,ind64
9=-1;
    short
ind650,ind651,ind652,ind653,ind654,ind655,ind656,ind657,ind658,ind65
9=-1;
    short
ind660,ind661,ind662,ind663,ind664,ind665,ind666,ind667,ind668,ind66
9=-1;
    short
ind670,ind671,ind672,ind673,ind674,ind675,ind676,ind677,ind678,ind67
9=-1;
    short
ind680,ind681,ind682,ind683,ind684,ind685,ind686,ind687,ind688,ind68
9=-1;
    short
ind690,ind691,ind692,ind693,ind694,ind695,ind696,ind697,ind698,ind69
9=-1;
    short
ind700,ind701,ind702,ind703,ind704,ind705,ind706,ind707,ind708,ind70
9=-1;
    exec sql end declare section;
#ifdef UF2DEBUG
  fprintf (myoutstream, "after declare\n");
  fflush(myoutstream);
#endif

    /* read inlistmax keys from the delete source file in */
    for (j=1; j <= inlistmax; j++)
    {
      fscanf (src_fh, "%s\n", keystr);
      key=atoi(keystr);

      switch(j)
      {
      case 1: inl001=key; ind001=0; break;
      case 2: inl002=key; ind002=0; break;
      case 3: inl003=key; ind003=0; break;
      case 4: inl004=key; ind004=0; break;
      case 5: inl005=key; ind005=0; break;
      case 6: inl006=key; ind006=0; break;
      case 7: inl007=key; ind007=0; break;
      case 8: inl008=key; ind008=0; break;
      case 9: inl009=key; ind009=0; break;
      case 10: inl010=key; ind010=0; break;
      case 11: inl011=key; ind011=0; break;
      case 12: inl012=key; ind012=0; break;
      case 13: inl013=key; ind013=0; break;
      case 14: inl014=key; ind014=0; break;
      case 15: inl015=key; ind015=0; break;
      case 16: inl016=key; ind016=0; break;
      case 17: inl017=key; ind017=0; break;
      case 18: inl018=key; ind018=0; break;
      case 19: inl019=key; ind019=0; break;
      case 20: inl020=key; ind020=0; break;
      case 21: inl021=key; ind021=0; break;
      case 22: inl022=key; ind022=0; break;
      case 23: inl023=key; ind023=0; break;
      case 24: inl024=key; ind024=0; break;
      case 25: inl025=key; ind025=0; break;
      case 26: inl026=key; ind026=0; break;
      case 27: inl027=key; ind027=0; break;
      case 28: inl028=key; ind028=0; break;
      case 29: inl029=key; ind029=0; break;
      case 30: inl030=key; ind030=0; break;
      case 31: inl031=key; ind031=0; break;
      case 32: inl032=key; ind032=0; break;
      case 33: inl033=key; ind033=0; break;
      case 34: inl034=key; ind034=0; break;
      case 35: inl035=key; ind035=0; break;
      case 36: inl036=key; ind036=0; break;
      case 37: inl037=key; ind037=0; break;
      case 38: inl038=key; ind038=0; break;
      case 39: inl039=key; ind039=0; break;
      case 40: inl040=key; ind040=0; break;
      case 41: inl041=key; ind041=0; break;
      case 42: inl042=key; ind042=0; break;
      case 43: inl043=key; ind043=0; break;
      case 44: inl044=key; ind044=0; break;
      case 45: inl045=key; ind045=0; break;
      case 46: inl046=key; ind046=0; break;
      case 47: inl047=key; ind047=0; break;
      case 48: inl048=key; ind048=0; break;
      case 49: inl049=key; ind049=0; break;
      case 50: inl050=key; ind050=0; break;
      case 51: inl051=key; ind051=0; break;
      case 52: inl052=key; ind052=0; break;
      case 53: inl053=key; ind053=0; break;
      case 54: inl054=key; ind054=0; break;
      case 55: inl055=key; ind055=0; break;
      case 56: inl056=key; ind056=0; break;
      case 57: inl057=key; ind057=0; break;
      case 58: inl058=key; ind058=0; break;
      case 59: inl059=key; ind059=0; break;
      case 60: inl060=key; ind060=0; break;
      case 61: inl061=key; ind061=0; break;
      case 62: inl062=key; ind062=0; break;
      case 63: inl063=key; ind063=0; break;
      case 64: inl064=key; ind064=0; break;
      case 65: inl065=key; ind065=0; break;
      case 66: inl066=key; ind066=0; break;
      case 67: inl067=key; ind067=0; break;
      case 68: inl068=key; ind068=0; break;
      case 69: inl069=key; ind069=0; break;
      case 70: inl070=key; ind070=0; break;
      case 71: inl071=key; ind071=0; break;
      case 72: inl072=key; ind072=0; break;
      case 73: inl073=key; ind073=0; break;
      case 74: inl074=key; ind074=0; break;
      case 75: inl075=key; ind075=0; break;
      case 76: inl076=key; ind076=0; break;
      case 77: inl077=key; ind077=0; break;
      case 78: inl078=key; ind078=0; break;
      case 79: inl079=key; ind079=0; break;
      case 80: inl080=key; ind080=0; break;
      case 81: inl081=key; ind081=0; break;
      case 82: inl082=key; ind082=0; break;
      case 83: inl083=key; ind083=0; break;
      case 84: inl084=key; ind084=0; break;
      case 85: inl085=key; ind085=0; break;
      case 86: inl086=key; ind086=0; break;
      case 87: inl087=key; ind087=0; break;
      case 88: inl088=key; ind088=0; break;
      case 89: inl089=key; ind089=0; break;
      case 90: inl090=key; ind090=0; break;
      case 91: inl091=key; ind091=0; break;
      case 92: inl092=key; ind092=0; break;
      case 93: inl093=key; ind093=0; break;
      case 94: inl094=key; ind094=0; break;
      case 95: inl095=key; ind095=0; break;
      case 96: inl096=key; ind096=0; break;
      case 97: inl097=key; ind097=0; break;
      case 98: inl098=key; ind098=0; break;
      case 99: inl099=key; ind099=0; break;
      case 100: inl100=key; ind100=0; break;
```

```
case 101: inl101=key; ind101=0; break;
case 102: inl102=key; ind102=0; break;
case 103: inl103=key; ind103=0; break;
case 104: inl104=key; ind104=0; break;
case 105: inl105=key; ind105=0; break;
case 106: inl106=key; ind106=0; break;
case 107: inl107=key; ind107=0; break;
case 108: inl108=key; ind108=0; break;
case 109: inl109=key; ind109=0; break;
case 110: inl110=key; ind110=0; break;
case 111: inl111=key; ind111=0; break;
case 112: inl112=key; ind112=0; break;
case 113: inl113=key; ind113=0; break;
case 114: inl114=key; ind114=0; break;
case 115: inl115=key; ind115=0; break;
case 116: inl116=key; ind116=0; break;
case 117: inl117=key; ind117=0; break;
case 118: inl118=key; ind118=0; break;
case 119: inl119=key; ind119=0; break;
case 120: inl120=key; ind120=0; break;
case 121: inl121=key; ind121=0; break;
case 122: inl122=key; ind122=0; break;
case 123: inl123=key; ind123=0; break;
case 124: inl124=key; ind124=0; break;
case 125: inl125=key; ind125=0; break;
case 126: inl126=key; ind126=0; break;
case 127: inl127=key; ind127=0; break;
case 128: inl128=key; ind128=0; break;
case 129: inl129=key; ind129=0; break;
case 130: inl130=key; ind130=0; break;
case 131: inl131=key; ind131=0; break;
case 132: inl132=key; ind132=0; break;
case 133: inl133=key; ind133=0; break;
case 134: inl134=key; ind134=0; break;
case 135: inl135=key; ind135=0; break;
case 136: inl136=key; ind136=0; break;
case 137: inl137=key; ind137=0; break;
case 138: inl138=key; ind138=0; break;
case 139: inl139=key; ind139=0; break;
case 140: inl140=key; ind140=0; break;
case 141: inl141=key; ind141=0; break;
case 142: inl142=key; ind142=0; break;
case 143: inl143=key; ind143=0; break;
case 144: inl144=key; ind144=0; break;
case 145: inl145=key; ind145=0; break;
case 146: inl146=key; ind146=0; break;
case 147: inl147=key; ind147=0; break;
case 148: inl148=key; ind148=0; break;
case 149: inl149=key; ind149=0; break;
case 150: inl150=key; ind150=0; break;
case 151: inl151=key; ind151=0; break;
case 152: inl152=key; ind152=0; break;
case 153: inl153=key; ind153=0; break;
case 154: inl154=key; ind154=0; break;
case 155: inl155=key; ind155=0; break;
case 156: inl156=key; ind156=0; break;
case 157: inl157=key; ind157=0; break;
case 158: inl158=key; ind158=0; break;
case 159: inl159=key; ind159=0; break;
case 160: inl160=key; ind160=0; break;
case 161: inl161=key; ind161=0; break;
case 162: inl162=key; ind162=0; break;
case 163: inl163=key; ind163=0; break;
case 164: inl164=key; ind164=0; break;
case 165: inl165=key; ind165=0; break;
case 166: inl166=key; ind166=0; break;
case 167: inl167=key; ind167=0; break;
case 168: inl168=key; ind168=0; break;
case 169: inl169=key; ind169=0; break;
case 170: inl170=key; ind170=0; break;
case 171: inl171=key; ind171=0; break;
case 172: inl172=key; ind172=0; break;
case 173: inl173=key; ind173=0; break;
case 174: inl174=key; ind174=0; break;
case 175: inl175=key; ind175=0; break;
case 176: inl176=key; ind176=0; break;
case 177: inl177=key; ind177=0; break;
case 178: inl178=key; ind178=0; break;
case 179: inl179=key; ind179=0; break;
case 180: inl180=key; ind180=0; break;
case 181: inl181=key; ind181=0; break;
case 182: inl182=key; ind182=0; break;
case 183: inl183=key; ind183=0; break;
case 184: inl184=key; ind184=0; break;
case 185: inl185=key; ind185=0; break;
case 186: inl186=key; ind186=0; break;
case 187: inl187=key; ind187=0; break;
case 188: inl188=key; ind188=0; break;
case 189: inl189=key; ind189=0; break;
case 190: inl190=key; ind190=0; break;
case 191: inl191=key; ind191=0; break;
case 192: inl192=key; ind192=0; break;
case 193: inl193=key; ind193=0; break;
case 194: inl194=key; ind194=0; break;
case 195: inl195=key; ind195=0; break;
case 196: inl196=key; ind196=0; break;
case 197: inl197=key; ind197=0; break;
case 198: inl198=key; ind198=0; break;
case 199: inl199=key; ind199=0; break;
case 200: inl200=key; ind200=0; break;
case 201: inl201=key; ind201=0; break;
case 202: inl202=key; ind202=0; break;
case 203: inl203=key; ind203=0; break;
case 204: inl204=key; ind204=0; break;
case 205: inl205=key; ind205=0; break;
case 206: inl206=key; ind206=0; break;
case 207: inl207=key; ind207=0; break;
case 208: inl208=key; ind208=0; break;
case 209: inl209=key; ind209=0; break;
case 210: inl210=key; ind210=0; break;
case 211: inl211=key; ind211=0; break;
case 212: inl212=key; ind212=0; break;
case 213: inl213=key; ind213=0; break;
case 214: inl214=key; ind214=0; break;
case 215: inl215=key; ind215=0; break;
case 216: inl216=key; ind216=0; break;
case 217: inl217=key; ind217=0; break;
case 218: inl218=key; ind218=0; break;
case 219: inl219=key; ind219=0; break;
case 220: inl220=key; ind220=0; break;
case 221: inl221=key; ind221=0; break;
case 222: inl222=key; ind222=0; break;
case 223: inl223=key; ind223=0; break;
case 224: inl224=key; ind224=0; break;
case 225: inl225=key; ind225=0; break;
case 226: inl226=key; ind226=0; break;
case 227: inl227=key; ind227=0; break;
case 228: inl228=key; ind228=0; break;
case 229: inl229=key; ind229=0; break;
case 230: inl230=key; ind230=0; break;
case 231: inl231=key; ind231=0; break;
case 232: inl232=key; ind232=0; break;
case 233: inl233=key; ind233=0; break;
case 234: inl234=key; ind234=0; break;
case 235: inl235=key; ind235=0; break;
case 236: inl236=key; ind236=0; break;
case 237: inl237=key; ind237=0; break;
case 238: inl238=key; ind238=0; break;
case 239: inl239=key; ind239=0; break;
case 240: inl240=key; ind240=0; break;
case 241: inl241=key; ind241=0; break;
case 242: inl242=key; ind242=0; break;
case 243: inl243=key; ind243=0; break;
case 244: inl244=key; ind244=0; break;
```

```
case 245: inl245=key; ind245=0; break;
case 246: inl246=key; ind246=0; break;
case 247: inl247=key; ind247=0; break;
case 248: inl248=key; ind248=0; break;
case 249: inl249=key; ind249=0; break;
case 250: inl250=key; ind250=0; break;
case 251: inl251=key; ind251=0; break;
case 252: inl252=key; ind252=0; break;
case 253: inl253=key; ind253=0; break;
case 254: inl254=key; ind254=0; break;
case 255: inl255=key; ind255=0; break;
case 256: inl256=key; ind256=0; break;
case 257: inl257=key; ind257=0; break;
case 258: inl258=key; ind258=0; break;
case 259: inl259=key; ind259=0; break;
case 260: inl260=key; ind260=0; break;
case 261: inl261=key; ind261=0; break;
case 262: inl262=key; ind262=0; break;
case 263: inl263=key; ind263=0; break;
case 264: inl264=key; ind264=0; break;
case 265: inl265=key; ind265=0; break;
case 266: inl266=key; ind266=0; break;
case 267: inl267=key; ind267=0; break;
case 268: inl268=key; ind268=0; break;
case 269: inl269=key; ind269=0; break;
case 270: inl270=key; ind270=0; break;
case 271: inl271=key; ind271=0; break;
case 272: inl272=key; ind272=0; break;
case 273: inl273=key; ind273=0; break;
case 274: inl274=key; ind274=0; break;
case 275: inl275=key; ind275=0; break;
case 276: inl276=key; ind276=0; break;
case 277: inl277=key; ind277=0; break;
case 278: inl278=key; ind278=0; break;
case 279: inl279=key; ind279=0; break;
case 280: inl280=key; ind280=0; break;
case 281: inl281=key; ind281=0; break;
case 282: inl282=key; ind282=0; break;
case 283: inl283=key; ind283=0; break;
case 284: inl284=key; ind284=0; break;
case 285: inl285=key; ind285=0; break;
case 286: inl286=key; ind286=0; break;
case 287: inl287=key; ind287=0; break;
case 288: inl288=key; ind288=0; break;
case 289: inl289=key; ind289=0; break;
case 290: inl290=key; ind290=0; break;
case 291: inl291=key; ind291=0; break;
case 292: inl292=key; ind292=0; break;
case 293: inl293=key; ind293=0; break;
case 294: inl294=key; ind294=0; break;
case 295: inl295=key; ind295=0; break;
case 296: inl296=key; ind296=0; break;
case 297: inl297=key; ind297=0; break;
case 298: inl298=key; ind298=0; break;
case 299: inl299=key; ind299=0; break;
case 300: inl300=key; ind300=0; break;
case 301: inl301=key; ind301=0; break;
case 302: inl302=key; ind302=0; break;
case 303: inl303=key; ind303=0; break;
case 304: inl304=key; ind304=0; break;
case 305: inl305=key; ind305=0; break;
case 306: inl306=key; ind306=0; break;
case 307: inl307=key; ind307=0; break;
case 308: inl308=key; ind308=0; break;
case 309: inl309=key; ind309=0; break;
case 310: inl310=key; ind310=0; break;
case 311: inl311=key; ind311=0; break;
case 312: inl312=key; ind312=0; break;
case 313: inl313=key; ind313=0; break;
case 314: inl314=key; ind314=0; break;
case 315: inl315=key; ind315=0; break;
case 316: inl316=key; ind316=0; break;
case 317: inl317=key; ind317=0; break;
case 318: inl318=key; ind318=0; break;
case 319: inl319=key; ind319=0; break;
case 320: inl320=key; ind320=0; break;
case 321: inl321=key; ind321=0; break;
case 322: inl322=key; ind322=0; break;
case 323: inl323=key; ind323=0; break;
case 324: inl324=key; ind324=0; break;
case 325: inl325=key; ind325=0; break;
case 326: inl326=key; ind326=0; break;
case 327: inl327=key; ind327=0; break;
case 328: inl328=key; ind328=0; break;
case 329: inl329=key; ind329=0; break;
case 330: inl330=key; ind330=0; break;
case 331: inl331=key; ind331=0; break;
case 332: inl332=key; ind332=0; break;
case 333: inl333=key; ind333=0; break;
case 334: inl334=key; ind334=0; break;
case 335: inl335=key; ind335=0; break;
case 336: inl336=key; ind336=0; break;
case 337: inl337=key; ind337=0; break;
case 338: inl338=key; ind338=0; break;
case 339: inl339=key; ind339=0; break;
case 340: inl340=key; ind340=0; break;
case 341: inl341=key; ind341=0; break;
case 342: inl342=key; ind342=0; break;
case 343: inl343=key; ind343=0; break;
case 344: inl344=key; ind344=0; break;
case 345: inl345=key; ind345=0; break;
case 346: inl346=key; ind346=0; break;
case 347: inl347=key; ind347=0; break;
case 348: inl348=key; ind348=0; break;
case 349: inl349=key; ind349=0; break;
case 350: inl350=key; ind350=0; break;
case 351: inl351=key; ind351=0; break;
case 352: inl352=key; ind352=0; break;
case 353: inl353=key; ind353=0; break;
case 354: inl354=key; ind354=0; break;
case 355: inl355=key; ind355=0; break;
case 356: inl356=key; ind356=0; break;
case 357: inl357=key; ind357=0; break;
case 358: inl358=key; ind358=0; break;
case 359: inl359=key; ind359=0; break;
case 360: inl360=key; ind360=0; break;
case 361: inl361=key; ind361=0; break;
case 362: inl362=key; ind362=0; break;
case 363: inl363=key; ind363=0; break;
case 364: inl364=key; ind364=0; break;
case 365: inl365=key; ind365=0; break;
case 366: inl366=key; ind366=0; break;
case 367: inl367=key; ind367=0; break;
case 368: inl368=key; ind368=0; break;
case 369: inl369=key; ind369=0; break;
case 370: inl370=key; ind370=0; break;
case 371: inl371=key; ind371=0; break;
case 372: inl372=key; ind372=0; break;
case 373: inl373=key; ind373=0; break;
case 374: inl374=key; ind374=0; break;
case 375: inl375=key; ind375=0; break;
case 376: inl376=key; ind376=0; break;
case 377: inl377=key; ind377=0; break;
case 378: inl378=key; ind378=0; break;
case 379: inl379=key; ind379=0; break;
case 380: inl380=key; ind380=0; break;
case 381: inl381=key; ind381=0; break;
case 382: inl382=key; ind382=0; break;
case 383: inl383=key; ind383=0; break;
case 384: inl384=key; ind384=0; break;
case 385: inl385=key; ind385=0; break;
case 386: inl386=key; ind386=0; break;
case 387: inl387=key; ind387=0; break;
case 388: inl388=key; ind388=0; break;
```

```
case 389: inl389=key; ind389=0; break;
case 390: inl390=key; ind390=0; break;
case 391: inl391=key; ind391=0; break;
case 392: inl392=key; ind392=0; break;
case 393: inl393=key; ind393=0; break;
case 394: inl394=key; ind394=0; break;
case 395: inl395=key; ind395=0; break;
case 396: inl396=key; ind396=0; break;
case 397: inl397=key; ind397=0; break;
case 398: inl398=key; ind398=0; break;
case 399: inl399=key; ind399=0; break;
case 400: inl400=key; ind400=0; break;
case 401: inl401=key; ind401=0; break;
case 402: inl402=key; ind402=0; break;
case 403: inl403=key; ind403=0; break;
case 404: inl404=key; ind404=0; break;
case 405: inl405=key; ind405=0; break;
case 406: inl406=key; ind406=0; break;
case 407: inl407=key; ind407=0; break;
case 408: inl408=key; ind408=0; break;
case 409: inl409=key; ind409=0; break;
case 410: inl410=key; ind410=0; break;
case 411: inl411=key; ind411=0; break;
case 412: inl412=key; ind412=0; break;
case 413: inl413=key; ind413=0; break;
case 414: inl414=key; ind414=0; break;
case 415: inl415=key; ind415=0; break;
case 416: inl416=key; ind416=0; break;
case 417: inl417=key; ind417=0; break;
case 418: inl418=key; ind418=0; break;
case 419: inl419=key; ind419=0; break;
case 420: inl420=key; ind420=0; break;
case 421: inl421=key; ind421=0; break;
case 422: inl422=key; ind422=0; break;
case 423: inl423=key; ind423=0; break;
case 424: inl424=key; ind424=0; break;
case 425: inl425=key; ind425=0; break;
case 426: inl426=key; ind426=0; break;
case 427: inl427=key; ind427=0; break;
case 428: inl428=key; ind428=0; break;
case 429: inl429=key; ind429=0; break;
case 430: inl430=key; ind430=0; break;
case 431: inl431=key; ind431=0; break;
case 432: inl432=key; ind432=0; break;
case 433: inl433=key; ind433=0; break;
case 434: inl434=key; ind434=0; break;
case 435: inl435=key; ind435=0; break;
case 436: inl436=key; ind436=0; break;
case 437: inl437=key; ind437=0; break;
case 438: inl438=key; ind438=0; break;
case 439: inl439=key; ind439=0; break;
case 440: inl440=key; ind440=0; break;
case 441: inl441=key; ind441=0; break;
case 442: inl442=key; ind442=0; break;
case 443: inl443=key; ind443=0; break;
case 444: inl444=key; ind444=0; break;
case 445: inl445=key; ind445=0; break;
case 446: inl446=key; ind446=0; break;
case 447: inl447=key; ind447=0; break;
case 448: inl448=key; ind448=0; break;
case 449: inl449=key; ind449=0; break;
case 450: inl450=key; ind450=0; break;
case 451: inl451=key; ind451=0; break;
case 452: inl452=key; ind452=0; break;
case 453: inl453=key; ind453=0; break;
case 454: inl454=key; ind454=0; break;
case 455: inl455=key; ind455=0; break;
case 456: inl456=key; ind456=0; break;
case 457: inl457=key; ind457=0; break;
case 458: inl458=key; ind458=0; break;
case 459: inl459=key; ind459=0; break;
case 460: inl460=key; ind460=0; break;
case 461: inl461=key; ind461=0; break;
case 462: inl462=key; ind462=0; break;
case 463: inl463=key; ind463=0; break;
case 464: inl464=key; ind464=0; break;
case 465: inl465=key; ind465=0; break;
case 466: inl466=key; ind466=0; break;
case 467: inl467=key; ind467=0; break;
case 468: inl468=key; ind468=0; break;
case 469: inl469=key; ind469=0; break;
case 470: inl470=key; ind470=0; break;
case 471: inl471=key; ind471=0; break;
case 472: inl472=key; ind472=0; break;
case 473: inl473=key; ind473=0; break;
case 474: inl474=key; ind474=0; break;
case 475: inl475=key; ind475=0; break;
case 476: inl476=key; ind476=0; break;
case 477: inl477=key; ind477=0; break;
case 478: inl478=key; ind478=0; break;
case 479: inl479=key; ind479=0; break;
case 480: inl480=key; ind480=0; break;
case 481: inl481=key; ind481=0; break;
case 482: inl482=key; ind482=0; break;
case 483: inl483=key; ind483=0; break;
case 484: inl484=key; ind484=0; break;
case 485: inl485=key; ind485=0; break;
case 486: inl486=key; ind486=0; break;
case 487: inl487=key; ind487=0; break;
case 488: inl488=key; ind488=0; break;
case 489: inl489=key; ind489=0; break;
case 490: inl490=key; ind490=0; break;
case 491: inl491=key; ind491=0; break;
case 492: inl492=key; ind492=0; break;
case 493: inl493=key; ind493=0; break;
case 494: inl494=key; ind494=0; break;
case 495: inl495=key; ind495=0; break;
case 496: inl496=key; ind496=0; break;
case 497: inl497=key; ind497=0; break;
case 498: inl498=key; ind498=0; break;
case 499: inl499=key; ind499=0; break;
case 500: inl500=key; ind500=0; break;
case 501: inl501=key; ind501=0; break;
case 502: inl502=key; ind502=0; break;
case 503: inl503=key; ind503=0; break;
case 504: inl504=key; ind504=0; break;
case 505: inl505=key; ind505=0; break;
case 506: inl506=key; ind506=0; break;
case 507: inl507=key; ind507=0; break;
case 508: inl508=key; ind508=0; break;
case 509: inl509=key; ind509=0; break;
case 510: inl510=key; ind510=0; break;
case 511: inl511=key; ind511=0; break;
case 512: inl512=key; ind512=0; break;
case 513: inl513=key; ind513=0; break;
case 514: inl514=key; ind514=0; break;
case 515: inl515=key; ind515=0; break;
case 516: inl516=key; ind516=0; break;
case 517: inl517=key; ind517=0; break;
case 518: inl518=key; ind518=0; break;
case 519: inl519=key; ind519=0; break;
case 520: inl520=key; ind520=0; break;
case 521: inl521=key; ind521=0; break;
case 522: inl522=key; ind522=0; break;
case 523: inl523=key; ind523=0; break;
case 524: inl524=key; ind524=0; break;
case 525: inl525=key; ind525=0; break;
case 526: inl526=key; ind526=0; break;
case 527: inl527=key; ind527=0; break;
case 528: inl528=key; ind528=0; break;
case 529: inl529=key; ind529=0; break;
case 530: inl530=key; ind530=0; break;
case 531: inl531=key; ind531=0; break;
case 532: inl532=key; ind532=0; break;
```

```
case 533: inl533=key; ind533=0; break;
case 534: inl534=key; ind534=0; break;
case 535: inl535=key; ind535=0; break;
case 536: inl536=key; ind536=0; break;
case 537: inl537=key; ind537=0; break;
case 538: inl538=key; ind538=0; break;
case 539: inl539=key; ind539=0; break;
case 540: inl540=key; ind540=0; break;
case 541: inl541=key; ind541=0; break;
case 542: inl542=key; ind542=0; break;
case 543: inl543=key; ind543=0; break;
case 544: inl544=key; ind544=0; break;
case 545: inl545=key; ind545=0; break;
case 546: inl546=key; ind546=0; break;
case 547: inl547=key; ind547=0; break;
case 548: inl548=key; ind548=0; break;
case 549: inl549=key; ind549=0; break;
case 550: inl550=key; ind550=0; break;
case 551: inl551=key; ind551=0; break;
case 552: inl552=key; ind552=0; break;
case 553: inl553=key; ind553=0; break;
case 554: inl554=key; ind554=0; break;
case 555: inl555=key; ind555=0; break;
case 556: inl556=key; ind556=0; break;
case 557: inl557=key; ind557=0; break;
case 558: inl558=key; ind558=0; break;
case 559: inl559=key; ind559=0; break;
case 560: inl560=key; ind560=0; break;
case 561: inl561=key; ind561=0; break;
case 562: inl562=key; ind562=0; break;
case 563: inl563=key; ind563=0; break;
case 564: inl564=key; ind564=0; break;
case 565: inl565=key; ind565=0; break;
case 566: inl566=key; ind566=0; break;
case 567: inl567=key; ind567=0; break;
case 568: inl568=key; ind568=0; break;
case 569: inl569=key; ind569=0; break;
case 570: inl570=key; ind570=0; break;
case 571: inl571=key; ind571=0; break;
case 572: inl572=key; ind572=0; break;
case 573: inl573=key; ind573=0; break;
case 574: inl574=key; ind574=0; break;
case 575: inl575=key; ind575=0; break;
case 576: inl576=key; ind576=0; break;
case 577: inl577=key; ind577=0; break;
case 578: inl578=key; ind578=0; break;
case 579: inl579=key; ind579=0; break;
case 580: inl580=key; ind580=0; break;
case 581: inl581=key; ind581=0; break;
case 582: inl582=key; ind582=0; break;
case 583: inl583=key; ind583=0; break;
case 584: inl584=key; ind584=0; break;
case 585: inl585=key; ind585=0; break;
case 586: inl586=key; ind586=0; break;
case 587: inl587=key; ind587=0; break;
case 588: inl588=key; ind588=0; break;
case 589: inl589=key; ind589=0; break;
case 590: inl590=key; ind590=0; break;
case 591: inl591=key; ind591=0; break;
case 592: inl592=key; ind592=0; break;
case 593: inl593=key; ind593=0; break;
case 594: inl594=key; ind594=0; break;
case 595: inl595=key; ind595=0; break;
case 596: inl596=key; ind596=0; break;
case 597: inl597=key; ind597=0; break;
case 598: inl598=key; ind598=0; break;
case 599: inl599=key; ind599=0; break;
case 600: inl600=key; ind600=0; break;
case 601: inl601=key; ind601=0; break;
case 602: inl602=key; ind602=0; break;
case 603: inl603=key; ind603=0; break;
case 604: inl604=key; ind604=0; break;
case 605: inl605=key; ind605=0; break;
case 606: inl606=key; ind606=0; break;
case 607: inl607=key; ind607=0; break;
case 608: inl608=key; ind608=0; break;
case 609: inl609=key; ind609=0; break;
case 610: inl610=key; ind610=0; break;
case 611: inl611=key; ind611=0; break;
case 612: inl612=key; ind612=0; break;
case 613: inl613=key; ind613=0; break;
case 614: inl614=key; ind614=0; break;
case 615: inl615=key; ind615=0; break;
case 616: inl616=key; ind616=0; break;
case 617: inl617=key; ind617=0; break;
case 618: inl618=key; ind618=0; break;
case 619: inl619=key; ind619=0; break;
case 620: inl620=key; ind620=0; break;
case 621: inl621=key; ind621=0; break;
case 622: inl622=key; ind622=0; break;
case 623: inl623=key; ind623=0; break;
case 624: inl624=key; ind624=0; break;
case 625: inl625=key; ind625=0; break;
case 626: inl626=key; ind626=0; break;
case 627: inl627=key; ind627=0; break;
case 628: inl628=key; ind628=0; break;
case 629: inl629=key; ind629=0; break;
case 630: inl630=key; ind630=0; break;
case 631: inl631=key; ind631=0; break;
case 632: inl632=key; ind632=0; break;
case 633: inl633=key; ind633=0; break;
case 634: inl634=key; ind634=0; break;
case 635: inl635=key; ind635=0; break;
case 636: inl636=key; ind636=0; break;
case 637: inl637=key; ind637=0; break;
case 638: inl638=key; ind638=0; break;
case 639: inl639=key; ind639=0; break;
case 640: inl640=key; ind640=0; break;
case 641: inl641=key; ind641=0; break;
case 642: inl642=key; ind642=0; break;
case 643: inl643=key; ind643=0; break;
case 644: inl644=key; ind644=0; break;
case 645: inl645=key; ind645=0; break;
case 646: inl646=key; ind646=0; break;
case 647: inl647=key; ind647=0; break;
case 648: inl648=key; ind648=0; break;
case 649: inl649=key; ind649=0; break;
case 650: inl650=key; ind650=0; break;
case 651: inl651=key; ind651=0; break;
case 652: inl652=key; ind652=0; break;
case 653: inl653=key; ind653=0; break;
case 654: inl654=key; ind654=0; break;
case 655: inl655=key; ind655=0; break;
case 656: inl656=key; ind656=0; break;
case 657: inl657=key; ind657=0; break;
case 658: inl658=key; ind658=0; break;
case 659: inl659=key; ind659=0; break;
case 660: inl660=key; ind660=0; break;
case 661: inl661=key; ind661=0; break;
case 662: inl662=key; ind662=0; break;
case 663: inl663=key; ind663=0; break;
case 664: inl664=key; ind664=0; break;
case 665: inl665=key; ind665=0; break;
case 666: inl666=key; ind666=0; break;
case 667: inl667=key; ind667=0; break;
case 668: inl668=key; ind668=0; break;
case 669: inl669=key; ind669=0; break;
case 670: inl670=key; ind670=0; break;
case 671: inl671=key; ind671=0; break;
case 672: inl672=key; ind672=0; break;
case 673: inl673=key; ind673=0; break;
case 674: inl674=key; ind674=0; break;
case 675: inl675=key; ind675=0; break;
case 676: inl676=key; ind676=0; break;
```

```
        case 677: inl677=key; ind677=0; break;
        case 678: inl678=key; ind678=0; break;
        case 679: inl679=key; ind679=0; break;
        case 680: inl680=key; ind680=0; break;
        case 681: inl681=key; ind681=0; break;
        case 682: inl682=key; ind682=0; break;
        case 683: inl683=key; ind683=0; break;
        case 684: inl684=key; ind684=0; break;
        case 685: inl685=key; ind685=0; break;
        case 686: inl686=key; ind686=0; break;
        case 687: inl687=key; ind687=0; break;
        case 688: inl688=key; ind688=0; break;
        case 689: inl689=key; ind689=0; break;
        case 690: inl690=key; ind690=0; break;
        case 691: inl691=key; ind691=0; break;
        case 692: inl692=key; ind692=0; break;
        case 693: inl693=key; ind693=0; break;
        case 694: inl694=key; ind694=0; break;
        case 695: inl695=key; ind695=0; break;
        case 696: inl696=key; ind696=0; break;
        case 697: inl697=key; ind697=0; break;
        case 698: inl698=key; ind698=0; break;
        case 699: inl699=key; ind699=0; break;
        case 700: inl700=key; ind700=0; break;
        case 701: inl701=key; ind701=0; break;
        case 702: inl702=key; ind702=0; break;
        case 703: inl703=key; ind703=0; break;
        case 704: inl704=key; ind704=0; break;
        case 705: inl705=key; ind705=0; break;
        case 706: inl706=key; ind706=0; break;
        case 707: inl707=key; ind707=0; break;
        case 708: inl708=key; ind708=0; break;
        case 709: inl709=key; ind709=0; break;
        default:
          fprintf (myoutstream,
            "\nHit an invalid inlist value j=%d\n EXITING",j);
          fflush(myoutstream);
          rc = -1;
          goto UF2_conn_reset;

        } /* end of switch */

        if (feof (src_fh))
        {
          done = TRUE;
          fclose (src_fh);
          break;
        }
      } /* end of inlistmax loop */

      /* Set things up for the loop which will retry if there is a deadlock */
      /* wlc  062797 */
      sqlcode = SQL_RC_E911;
      maxwait = 1;
      rc = 0;

      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.ORDERS WHERE
NODENUMBER(O_ORDERKEY) = CURRENT NODE AND
          O_ORDERKEY IN (
 :inl001:ind001,
 :inl002:ind002,
 :inl003:ind003,
 :inl004:ind004,
 :inl005:ind005,
 :inl006:ind006,
 :inl007:ind007,
 :inl008:ind008,
 :inl009:ind009,
 :inl010:ind010,
 :inl011:ind011,
 :inl012:ind012,
 :inl013:ind013,
 :inl014:ind014,
 :inl015:ind015,
 :inl016:ind016,
 :inl017:ind017,
 :inl018:ind018,
 :inl019:ind019,
 :inl020:ind020,
 :inl021:ind021,
 :inl022:ind022,
 :inl023:ind023,
 :inl024:ind024,
 :inl025:ind025,
 :inl026:ind026,
 :inl027:ind027,
 :inl028:ind028,
 :inl029:ind029,
 :inl030:ind030,
 :inl031:ind031,
 :inl032:ind032,
 :inl033:ind033,
 :inl034:ind034,
 :inl035:ind035,
 :inl036:ind036,
 :inl037:ind037,
 :inl038:ind038,
 :inl039:ind039,
 :inl040:ind040,
 :inl041:ind041,
 :inl042:ind042,
 :inl043:ind043,
 :inl044:ind044,
 :inl045:ind045,
 :inl046:ind046,
 :inl047:ind047,
 :inl048:ind048,
 :inl049:ind049,
 :inl050:ind050,
 :inl051:ind051,
 :inl052:ind052,
 :inl053:ind053,
 :inl054:ind054,
 :inl055:ind055,
 :inl056:ind056,
 :inl057:ind057,
 :inl058:ind058,
 :inl059:ind059,
 :inl060:ind060,
 :inl061:ind061,
 :inl062:ind062,
 :inl063:ind063,
 :inl064:ind064,
 :inl065:ind065,
 :inl066:ind066,
 :inl067:ind067,
 :inl068:ind068,
 :inl069:ind069,
 :inl070:ind070,
 :inl071:ind071,
 :inl072:ind072,
 :inl073:ind073,
 :inl074:ind074,
 :inl075:ind075,
```

:inl076:ind076,
:inl077:ind077,
:inl078:ind078,
:inl079:ind079,
:inl080:ind080,
:inl081:ind081,
:inl082:ind082,
:inl083:ind083,
:inl084:ind084,
:inl085:ind085,
:inl086:ind086,
:inl087:ind087,
:inl088:ind088,
:inl089:ind089,
:inl090:ind090,
:inl091:ind091,
:inl092:ind092,
:inl093:ind093,
:inl094:ind094,
:inl095:ind095,
:inl096:ind096,
:inl097:ind097,
:inl098:ind098,
:inl099:ind099,
:inl100:ind100,
:inl101:ind101,
:inl102:ind102,
:inl103:ind103,
:inl104:ind104,
:inl105:ind105,
:inl106:ind106,
:inl107:ind107,
:inl108:ind108,
:inl109:ind109,
:inl110:ind110,
:inl111:ind111,
:inl112:ind112,
:inl113:ind113,
:inl114:ind114,
:inl115:ind115,
:inl116:ind116,
:inl117:ind117,
:inl118:ind118,
:inl119:ind119,
:inl120:ind120,
:inl121:ind121,
:inl122:ind122,
:inl123:ind123,
:inl124:ind124,
:inl125:ind125,
:inl126:ind126,
:inl127:ind127,
:inl128:ind128,
:inl129:ind129,
:inl130:ind130,
:inl131:ind131,
:inl132:ind132,
:inl133:ind133,
:inl134:ind134,
:inl135:ind135,
:inl136:ind136,
:inl137:ind137,
:inl138:ind138,
:inl139:ind139,
:inl140:ind140,
:inl141:ind141,
:inl142:ind142,
:inl143:ind143,
:inl144:ind144,
:inl145:ind145,
:inl146:ind146,
:inl147:ind147,
:inl148:ind148,
:inl149:ind149,
:inl150:ind150,
:inl151:ind151,
:inl152:ind152,
:inl153:ind153,
:inl154:ind154,
:inl155:ind155,
:inl156:ind156,
:inl157:ind157,
:inl158:ind158,
:inl159:ind159,
:inl160:ind160,
:inl161:ind161,
:inl162:ind162,
:inl163:ind163,
:inl164:ind164,
:inl165:ind165,
:inl166:ind166,
:inl167:ind167,
:inl168:ind168,
:inl169:ind169,
:inl170:ind170,
:inl171:ind171,
:inl172:ind172,
:inl173:ind173,
:inl174:ind174,
:inl175:ind175,
:inl176:ind176,
:inl177:ind177,
:inl178:ind178,
:inl179:ind179,
:inl180:ind180,
:inl181:ind181,
:inl182:ind182,
:inl183:ind183,
:inl184:ind184,
:inl185:ind185,
:inl186:ind186,
:inl187:ind187,
:inl188:ind188,
:inl189:ind189,
:inl190:ind190,
:inl191:ind191,
:inl192:ind192,
:inl193:ind193,
:inl194:ind194,
:inl195:ind195,
:inl196:ind196,
:inl197:ind197,
:inl198:ind198,
:inl199:ind199,
:inl200:ind200,
:inl201:ind201,
:inl202:ind202,
:inl203:ind203,
:inl204:ind204,
:inl205:ind205,
:inl206:ind206,
:inl207:ind207,
:inl208:ind208,
:inl209:ind209,
:inl210:ind210,
:inl211:ind211,
:inl212:ind212,
:inl213:ind213,
:inl214:ind214,
:inl215:ind215,
:inl216:ind216,
:inl217:ind217,
:inl218:ind218,
:inl219:ind219,

:inl220:ind220,
:inl221:ind221,
:inl222:ind222,
:inl223:ind223,
:inl224:ind224,
:inl225:ind225,
:inl226:ind226,
:inl227:ind227,
:inl228:ind228,
:inl229:ind229,
:inl230:ind230,
:inl231:ind231,
:inl232:ind232,
:inl233:ind233,
:inl234:ind234,
:inl235:ind235,
:inl236:ind236,
:inl237:ind237,
:inl238:ind238,
:inl239:ind239,
:inl240:ind240,
:inl241:ind241,
:inl242:ind242,
:inl243:ind243,
:inl244:ind244,
:inl245:ind245,
:inl246:ind246,
:inl247:ind247,
:inl248:ind248,
:inl249:ind249,
:inl250:ind250,
:inl251:ind251,
:inl252:ind252,
:inl253:ind253,
:inl254:ind254,
:inl255:ind255,
:inl256:ind256,
:inl257:ind257,
:inl258:ind258,
:inl259:ind259,
:inl260:ind260,
:inl261:ind261,
:inl262:ind262,
:inl263:ind263,
:inl264:ind264,
:inl265:ind265,
:inl266:ind266,
:inl267:ind267,
:inl268:ind268,
:inl269:ind269,
:inl270:ind270,
:inl271:ind271,
:inl272:ind272,
:inl273:ind273,
:inl274:ind274,
:inl275:ind275,
:inl276:ind276,
:inl277:ind277,
:inl278:ind278,
:inl279:ind279,
:inl280:ind280,
:inl281:ind281,
:inl282:ind282,
:inl283:ind283,
:inl284:ind284,
:inl285:ind285,
:inl286:ind286,
:inl287:ind287,
:inl288:ind288,
:inl289:ind289,
:inl290:ind290,
:inl291:ind291,
:inl292:ind292,
:inl293:ind293,
:inl294:ind294,
:inl295:ind295,
:inl296:ind296,
:inl297:ind297,
:inl298:ind298,
:inl299:ind299,
:inl300:ind300,
:inl301:ind301,
:inl302:ind302,
:inl303:ind303,
:inl304:ind304,
:inl305:ind305,
:inl306:ind306,
:inl307:ind307,
:inl308:ind308,
:inl309:ind309,
:inl310:ind310,
:inl311:ind311,
:inl312:ind312,
:inl313:ind313,
:inl314:ind314,
:inl315:ind315,
:inl316:ind316,
:inl317:ind317,
:inl318:ind318,
:inl319:ind319,
:inl320:ind320,
:inl321:ind321,
:inl322:ind322,
:inl323:ind323,
:inl324:ind324,
:inl325:ind325,
:inl326:ind326,
:inl327:ind327,
:inl328:ind328,
:inl329:ind329,
:inl330:ind330,
:inl331:ind331,
:inl332:ind332,
:inl333:ind333,
:inl334:ind334,
:inl335:ind335,
:inl336:ind336,
:inl337:ind337,
:inl338:ind338,
:inl339:ind339,
:inl340:ind340,
:inl341:ind341,
:inl342:ind342,
:inl343:ind343,
:inl344:ind344,
:inl345:ind345,
:inl346:ind346,
:inl347:ind347,
:inl348:ind348,
:inl349:ind349,
:inl350:ind350,
:inl351:ind351,
:inl352:ind352,
:inl353:ind353,
:inl354:ind354,
:inl355:ind355,
:inl356:ind356,
:inl357:ind357,
:inl358:ind358,
:inl359:ind359,
:inl360:ind360,
:inl361:ind361,
:inl362:ind362,
:inl363:ind363,

:inl364:ind364,
:inl365:ind365,
:inl366:ind366,
:inl367:ind367,
:inl368:ind368,
:inl369:ind369,
:inl370:ind370,
:inl371:ind371,
:inl372:ind372,
:inl373:ind373,
:inl374:ind374,
:inl375:ind375,
:inl376:ind376,
:inl377:ind377,
:inl378:ind378,
:inl379:ind379,
:inl380:ind380,
:inl381:ind381,
:inl382:ind382,
:inl383:ind383,
:inl384:ind384,
:inl385:ind385,
:inl386:ind386,
:inl387:ind387,
:inl388:ind388,
:inl389:ind389,
:inl390:ind390,
:inl391:ind391,
:inl392:ind392,
:inl393:ind393,
:inl394:ind394,
:inl395:ind395,
:inl396:ind396,
:inl397:ind397,
:inl398:ind398,
:inl399:ind399,
:inl400:ind400,
:inl401:ind401,
:inl402:ind402,
:inl403:ind403,
:inl404:ind404,
:inl405:ind405,
:inl406:ind406,
:inl407:ind407,
:inl408:ind408,
:inl409:ind409,
:inl410:ind410,
:inl411:ind411,
:inl412:ind412,
:inl413:ind413,
:inl414:ind414,
:inl415:ind415,
:inl416:ind416,
:inl417:ind417,
:inl418:ind418,
:inl419:ind419,
:inl420:ind420,
:inl421:ind421,
:inl422:ind422,
:inl423:ind423,
:inl424:ind424,
:inl425:ind425,
:inl426:ind426,
:inl427:ind427,
:inl428:ind428,
:inl429:ind429,
:inl430:ind430,
:inl431:ind431,
:inl432:ind432,
:inl433:ind433,
:inl434:ind434,
:inl435:ind435,
:inl436:ind436,
:inl437:ind437,
:inl438:ind438,
:inl439:ind439,
:inl440:ind440,
:inl441:ind441,
:inl442:ind442,
:inl443:ind443,
:inl444:ind444,
:inl445:ind445,
:inl446:ind446,
:inl447:ind447,
:inl448:ind448,
:inl449:ind449,
:inl450:ind450,
:inl451:ind451,
:inl452:ind452,
:inl453:ind453,
:inl454:ind454,
:inl455:ind455,
:inl456:ind456,
:inl457:ind457,
:inl458:ind458,
:inl459:ind459,
:inl460:ind460,
:inl461:ind461,
:inl462:ind462,
:inl463:ind463,
:inl464:ind464,
:inl465:ind465,
:inl466:ind466,
:inl467:ind467,
:inl468:ind468,
:inl469:ind469,
:inl470:ind470,
:inl471:ind471,
:inl472:ind472,
:inl473:ind473,
:inl474:ind474,
:inl475:ind475,
:inl476:ind476,
:inl477:ind477,
:inl478:ind478,
:inl479:ind479,
:inl480:ind480,
:inl481:ind481,
:inl482:ind482,
:inl483:ind483,
:inl484:ind484,
:inl485:ind485,
:inl486:ind486,
:inl487:ind487,
:inl488:ind488,
:inl489:ind489,
:inl490:ind490,
:inl491:ind491,
:inl492:ind492,
:inl493:ind493,
:inl494:ind494,
:inl495:ind495,
:inl496:ind496,
:inl497:ind497,
:inl498:ind498,
:inl499:ind499,
:inl500:ind500,
:inl501:ind501,
:inl502:ind502,
:inl503:ind503,
:inl504:ind504,
:inl505:ind505,
:inl506:ind506,
:inl507:ind507,

:inl508:ind508,
:inl509:ind509,
:inl510:ind510,
:inl511:ind511,
:inl512:ind512,
:inl513:ind513,
:inl514:ind514,
:inl515:ind515,
:inl516:ind516,
:inl517:ind517,
:inl518:ind518,
:inl519:ind519,
:inl520:ind520,
:inl521:ind521,
:inl522:ind522,
:inl523:ind523,
:inl524:ind524,
:inl525:ind525,
:inl526:ind526,
:inl527:ind527,
:inl528:ind528,
:inl529:ind529,
:inl530:ind530,
:inl531:ind531,
:inl532:ind532,
:inl533:ind533,
:inl534:ind534,
:inl535:ind535,
:inl536:ind536,
:inl537:ind537,
:inl538:ind538,
:inl539:ind539,
:inl540:ind540,
:inl541:ind541,
:inl542:ind542,
:inl543:ind543,
:inl544:ind544,
:inl545:ind545,
:inl546:ind546,
:inl547:ind547,
:inl548:ind548,
:inl549:ind549,
:inl550:ind550,
:inl551:ind551,
:inl552:ind552,
:inl553:ind553,
:inl554:ind554,
:inl555:ind555,
:inl556:ind556,
:inl557:ind557,
:inl558:ind558,
:inl559:ind559,
:inl560:ind560,
:inl561:ind561,
:inl562:ind562,
:inl563:ind563,
:inl564:ind564,
:inl565:ind565,
:inl566:ind566,
:inl567:ind567,
:inl568:ind568,
:inl569:ind569,
:inl570:ind570,
:inl571:ind571,
:inl572:ind572,
:inl573:ind573,
:inl574:ind574,
:inl575:ind575,
:inl576:ind576,
:inl577:ind577,
:inl578:ind578,
:inl579:ind579,
:inl580:ind580,
:inl581:ind581,
:inl582:ind582,
:inl583:ind583,
:inl584:ind584,
:inl585:ind585,
:inl586:ind586,
:inl587:ind587,
:inl588:ind588,
:inl589:ind589,
:inl590:ind590,
:inl591:ind591,
:inl592:ind592,
:inl593:ind593,
:inl594:ind594,
:inl595:ind595,
:inl596:ind596,
:inl597:ind597,
:inl598:ind598,
:inl599:ind599,
:inl600:ind600,
:inl601:ind601,
:inl602:ind602,
:inl603:ind603,
:inl604:ind604,
:inl605:ind605,
:inl606:ind606,
:inl607:ind607,
:inl608:ind608,
:inl609:ind609,
:inl610:ind610,
:inl611:ind611,
:inl612:ind612,
:inl613:ind613,
:inl614:ind614,
:inl615:ind615,
:inl616:ind616,
:inl617:ind617,
:inl618:ind618,
:inl619:ind619,
:inl620:ind620,
:inl621:ind621,
:inl622:ind622,
:inl623:ind623,
:inl624:ind624,
:inl625:ind625,
:inl626:ind626,
:inl627:ind627,
:inl628:ind628,
:inl629:ind629,
:inl630:ind630,
:inl631:ind631,
:inl632:ind632,
:inl633:ind633,
:inl634:ind634,
:inl635:ind635,
:inl636:ind636,
:inl637:ind637,
:inl638:ind638,
:inl639:ind639,
:inl640:ind640,
:inl641:ind641,
:inl642:ind642,
:inl643:ind643,
:inl644:ind644,
:inl645:ind645,
:inl646:ind646,
:inl647:ind647,
:inl648:ind648,
:inl649:ind649,
:inl650:ind650,
:inl651:ind651,

```
:inl652:ind652,
:inl653:ind653,
:inl654:ind654,
:inl655:ind655,
:inl656:ind656,
:inl657:ind657,
:inl658:ind658,
:inl659:ind659,
:inl660:ind660,
:inl661:ind661,
:inl662:ind662,
:inl663:ind663,
:inl664:ind664,
:inl665:ind665,
:inl666:ind666,
:inl667:ind667,
:inl668:ind668,
:inl669:ind669,
:inl670:ind670,
:inl671:ind671,
:inl672:ind672,
:inl673:ind673,
:inl674:ind674,
:inl675:ind675,
:inl676:ind676,
:inl677:ind677,
:inl678:ind678,
:inl679:ind679,
:inl680:ind680,
:inl681:ind681,
:inl682:ind682,
:inl683:ind683,
:inl684:ind684,
:inl685:ind685,
:inl686:ind686,
:inl687:ind687,
:inl688:ind688,
:inl689:ind689,
:inl690:ind690,
:inl691:ind691,
:inl692:ind692,
:inl693:ind693,
:inl694:ind694,
:inl695:ind695,
:inl696:ind696,
:inl697:ind697,
:inl698:ind698,
:inl699:ind699,
:inl700:ind700,
:inl701:ind701,
:inl702:ind702,
:inl703:ind703,
:inl704:ind704,
:inl705:ind705,
:inl706:ind706,
:inl707:ind707,
:inl708:ind708,
:inl709:ind709
);

        sqlcode = error_check();    /* Don't bother printing the SQL0100W
                                       error from going through the loop
                                       above one extra time */

        if (sqlcode == SQL_RC_E911)
        {         /* we've hit a deadlock */
          fprintf (myoutstream, "\nA deadlock has been
detected...Retrying...\n");
          SleepSome(UF_DEADLOCK_SLEEP);
          maxwait++;      /* jen DEADLOCK */
        }
        else if (sqlca.sqlcode < 0)
        {
          fprintf (myoutstream,"\n%s\n", stmt_str.data);
          fprintf (myoutstream,"\nsqlcode %d occurred deleting from
TPCD.ORDERS\n", sqlcode);
          if (sqlca.sqlerrml)
          {
            char *tokptr;
            int  tokl;
            fprintf (myoutstream,"\n     SQLCA:   tokens:\n");
            tokptr=strtok(sqlca.sqlerrmc, "\xff");
            while (  tokptr                          &&
                   ( (tokl = (sizeof(sqlca.sqlerrmc) - (tokptr-sqlca.sqlerrmc)))
> 0)
                   )
            {
              fprintf (myoutstream,"%.*s\n", tokl, tokptr);
              tokptr=strtok(NULL, "\xff");
            }
          }
          fprintf (myoutstream,"\n     SQLCA:   errp= %.8s, errd 1-6= %d %d
%d %d %d %d\n",
                 sqlca.sqlerrp, sqlca.sqlerrd[0], sqlca.sqlerrd[1],
sqlca.sqlerrd[2],
                 sqlca.sqlerrd[3], sqlca.sqlerrd[4], sqlca.sqlerrd[5]);

          fprintf (myoutstream,
                 "for update pair number %d node %d chunk %d..Exiting\n",
                 updatePair, i,deleteChunk);
          rc=-1;
        }
        else
        {
          /* report the number of row deleted */
          fprintf(myoutstream, "%ld rows deleted from TPCD.ORDERS at
%18.18s\n",
                 sqlca.sqlerrd[2],
                 get_time_stamp(T_STAMP_FORM_1,(time_t)NULL));

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

          /* delete the lineitems now */

          EXEC SQL DELETE FROM TPCD.LINEITEM WHERE
NODENUMBER(L_ORDERKEY) = CURRENT NODE AND
              L_ORDERKEY IN (
 :inl001:ind001,
 :inl002:ind002,
 :inl003:ind003,
 :inl004:ind004,
 :inl005:ind005,
 :inl006:ind006,
 :inl007:ind007,
 :inl008:ind008,
 :inl009:ind009,
 :inl010:ind010,
 :inl011:ind011,
 :inl012:ind012,
 :inl013:ind013,
 :inl014:ind014,
 :inl015:ind015,
 :inl016:ind016,
 :inl017:ind017,
 :inl018:ind018,
 :inl019:ind019,
 :inl020:ind020,
 :inl021:ind021,
 :inl022:ind022,
```

:inl023:ind023,
:inl024:ind024,
:inl025:ind025,
:inl026:ind026,
:inl027:ind027,
:inl028:ind028,
:inl029:ind029,
:inl030:ind030,
:inl031:ind031,
:inl032:ind032,
:inl033:ind033,
:inl034:ind034,
:inl035:ind035,
:inl036:ind036,
:inl037:ind037,
:inl038:ind038,
:inl039:ind039,
:inl040:ind040,
:inl041:ind041,
:inl042:ind042,
:inl043:ind043,
:inl044:ind044,
:inl045:ind045,
:inl046:ind046,
:inl047:ind047,
:inl048:ind048,
:inl049:ind049,
:inl050:ind050,
:inl051:ind051,
:inl052:ind052,
:inl053:ind053,
:inl054:ind054,
:inl055:ind055,
:inl056:ind056,
:inl057:ind057,
:inl058:ind058,
:inl059:ind059,
:inl060:ind060,
:inl061:ind061,
:inl062:ind062,
:inl063:ind063,
:inl064:ind064,
:inl065:ind065,
:inl066:ind066,
:inl067:ind067,
:inl068:ind068,
:inl069:ind069,
:inl070:ind070,
:inl071:ind071,
:inl072:ind072,
:inl073:ind073,
:inl074:ind074,
:inl075:ind075,
:inl076:ind076,
:inl077:ind077,
:inl078:ind078,
:inl079:ind079,
:inl080:ind080,
:inl081:ind081,
:inl082:ind082,
:inl083:ind083,
:inl084:ind084,
:inl085:ind085,
:inl086:ind086,
:inl087:ind087,
:inl088:ind088,
:inl089:ind089,
:inl090:ind090,
:inl091:ind091,
:inl092:ind092,
:inl093:ind093,
:inl094:ind094,
:inl095:ind095,
:inl096:ind096,
:inl097:ind097,
:inl098:ind098,
:inl099:ind099,
:inl100:ind100,
:inl101:ind101,
:inl102:ind102,
:inl103:ind103,
:inl104:ind104,
:inl105:ind105,
:inl106:ind106,
:inl107:ind107,
:inl108:ind108,
:inl109:ind109,
:inl110:ind110,
:inl111:ind111,
:inl112:ind112,
:inl113:ind113,
:inl114:ind114,
:inl115:ind115,
:inl116:ind116,
:inl117:ind117,
:inl118:ind118,
:inl119:ind119,
:inl120:ind120,
:inl121:ind121,
:inl122:ind122,
:inl123:ind123,
:inl124:ind124,
:inl125:ind125,
:inl126:ind126,
:inl127:ind127,
:inl128:ind128,
:inl129:ind129,
:inl130:ind130,
:inl131:ind131,
:inl132:ind132,
:inl133:ind133,
:inl134:ind134,
:inl135:ind135,
:inl136:ind136,
:inl137:ind137,
:inl138:ind138,
:inl139:ind139,
:inl140:ind140,
:inl141:ind141,
:inl142:ind142,
:inl143:ind143,
:inl144:ind144,
:inl145:ind145,
:inl146:ind146,
:inl147:ind147,
:inl148:ind148,
:inl149:ind149,
:inl150:ind150,
:inl151:ind151,
:inl152:ind152,
:inl153:ind153,
:inl154:ind154,
:inl155:ind155,
:inl156:ind156,
:inl157:ind157,
:inl158:ind158,
:inl159:ind159,
:inl160:ind160,
:inl161:ind161,
:inl162:ind162,
:inl163:ind163,
:inl164:ind164,
:inl165:ind165,
:inl166:ind166,

:inl167:ind167,
:inl168:ind168,
:inl169:ind169,
:inl170:ind170,
:inl171:ind171,
:inl172:ind172,
:inl173:ind173,
:inl174:ind174,
:inl175:ind175,
:inl176:ind176,
:inl177:ind177,
:inl178:ind178,
:inl179:ind179,
:inl180:ind180,
:inl181:ind181,
:inl182:ind182,
:inl183:ind183,
:inl184:ind184,
:inl185:ind185,
:inl186:ind186,
:inl187:ind187,
:inl188:ind188,
:inl189:ind189,
:inl190:ind190,
:inl191:ind191,
:inl192:ind192,
:inl193:ind193,
:inl194:ind194,
:inl195:ind195,
:inl196:ind196,
:inl197:ind197,
:inl198:ind198,
:inl199:ind199,
:inl200:ind200,
:inl201:ind201,
:inl202:ind202,
:inl203:ind203,
:inl204:ind204,
:inl205:ind205,
:inl206:ind206,
:inl207:ind207,
:inl208:ind208,
:inl209:ind209,
:inl210:ind210,
:inl211:ind211,
:inl212:ind212,
:inl213:ind213,
:inl214:ind214,
:inl215:ind215,
:inl216:ind216,
:inl217:ind217,
:inl218:ind218,
:inl219:ind219,
:inl220:ind220,
:inl221:ind221,
:inl222:ind222,
:inl223:ind223,
:inl224:ind224,
:inl225:ind225,
:inl226:ind226,
:inl227:ind227,
:inl228:ind228,
:inl229:ind229,
:inl230:ind230,
:inl231:ind231,
:inl232:ind232,
:inl233:ind233,
:inl234:ind234,
:inl235:ind235,
:inl236:ind236,
:inl237:ind237,
:inl238:ind238,
:inl239:ind239,
:inl240:ind240,
:inl241:ind241,
:inl242:ind242,
:inl243:ind243,
:inl244:ind244,
:inl245:ind245,
:inl246:ind246,
:inl247:ind247,
:inl248:ind248,
:inl249:ind249,
:inl250:ind250,
:inl251:ind251,
:inl252:ind252,
:inl253:ind253,
:inl254:ind254,
:inl255:ind255,
:inl256:ind256,
:inl257:ind257,
:inl258:ind258,
:inl259:ind259,
:inl260:ind260,
:inl261:ind261,
:inl262:ind262,
:inl263:ind263,
:inl264:ind264,
:inl265:ind265,
:inl266:ind266,
:inl267:ind267,
:inl268:ind268,
:inl269:ind269,
:inl270:ind270,
:inl271:ind271,
:inl272:ind272,
:inl273:ind273,
:inl274:ind274,
:inl275:ind275,
:inl276:ind276,
:inl277:ind277,
:inl278:ind278,
:inl279:ind279,
:inl280:ind280,
:inl281:ind281,
:inl282:ind282,
:inl283:ind283,
:inl284:ind284,
:inl285:ind285,
:inl286:ind286,
:inl287:ind287,
:inl288:ind288,
:inl289:ind289,
:inl290:ind290,
:inl291:ind291,
:inl292:ind292,
:inl293:ind293,
:inl294:ind294,
:inl295:ind295,
:inl296:ind296,
:inl297:ind297,
:inl298:ind298,
:inl299:ind299,
:inl300:ind300,
:inl301:ind301,
:inl302:ind302,
:inl303:ind303,
:inl304:ind304,
:inl305:ind305,
:inl306:ind306,
:inl307:ind307,
:inl308:ind308,
:inl309:ind309,
:inl310:ind310,

:inl311:ind311,
:inl312:ind312,
:inl313:ind313,
:inl314:ind314,
:inl315:ind315,
:inl316:ind316,
:inl317:ind317,
:inl318:ind318,
:inl319:ind319,
:inl320:ind320,
:inl321:ind321,
:inl322:ind322,
:inl323:ind323,
:inl324:ind324,
:inl325:ind325,
:inl326:ind326,
:inl327:ind327,
:inl328:ind328,
:inl329:ind329,
:inl330:ind330,
:inl331:ind331,
:inl332:ind332,
:inl333:ind333,
:inl334:ind334,
:inl335:ind335,
:inl336:ind336,
:inl337:ind337,
:inl338:ind338,
:inl339:ind339,
:inl340:ind340,
:inl341:ind341,
:inl342:ind342,
:inl343:ind343,
:inl344:ind344,
:inl345:ind345,
:inl346:ind346,
:inl347:ind347,
:inl348:ind348,
:inl349:ind349,
:inl350:ind350,
:inl351:ind351,
:inl352:ind352,
:inl353:ind353,
:inl354:ind354,
:inl355:ind355,
:inl356:ind356,
:inl357:ind357,
:inl358:ind358,
:inl359:ind359,
:inl360:ind360,
:inl361:ind361,
:inl362:ind362,
:inl363:ind363,
:inl364:ind364,
:inl365:ind365,
:inl366:ind366,
:inl367:ind367,
:inl368:ind368,
:inl369:ind369,
:inl370:ind370,
:inl371:ind371,
:inl372:ind372,
:inl373:ind373,
:inl374:ind374,
:inl375:ind375,
:inl376:ind376,
:inl377:ind377,
:inl378:ind378,
:inl379:ind379,
:inl380:ind380,
:inl381:ind381,
:inl382:ind382,
:inl383:ind383,
:inl384:ind384,
:inl385:ind385,
:inl386:ind386,
:inl387:ind387,
:inl388:ind388,
:inl389:ind389,
:inl390:ind390,
:inl391:ind391,
:inl392:ind392,
:inl393:ind393,
:inl394:ind394,
:inl395:ind395,
:inl396:ind396,
:inl397:ind397,
:inl398:ind398,
:inl399:ind399,
:inl400:ind400,
:inl401:ind401,
:inl402:ind402,
:inl403:ind403,
:inl404:ind404,
:inl405:ind405,
:inl406:ind406,
:inl407:ind407,
:inl408:ind408,
:inl409:ind409,
:inl410:ind410,
:inl411:ind411,
:inl412:ind412,
:inl413:ind413,
:inl414:ind414,
:inl415:ind415,
:inl416:ind416,
:inl417:ind417,
:inl418:ind418,
:inl419:ind419,
:inl420:ind420,
:inl421:ind421,
:inl422:ind422,
:inl423:ind423,
:inl424:ind424,
:inl425:ind425,
:inl426:ind426,
:inl427:ind427,
:inl428:ind428,
:inl429:ind429,
:inl430:ind430,
:inl431:ind431,
:inl432:ind432,
:inl433:ind433,
:inl434:ind434,
:inl435:ind435,
:inl436:ind436,
:inl437:ind437,
:inl438:ind438,
:inl439:ind439,
:inl440:ind440,
:inl441:ind441,
:inl442:ind442,
:inl443:ind443,
:inl444:ind444,
:inl445:ind445,
:inl446:ind446,
:inl447:ind447,
:inl448:ind448,
:inl449:ind449,
:inl450:ind450,
:inl451:ind451,
:inl452:ind452,
:inl453:ind453,
:inl454:ind454,

:inl455:ind455,
:inl456:ind456,
:inl457:ind457,
:inl458:ind458,
:inl459:ind459,
:inl460:ind460,
:inl461:ind461,
:inl462:ind462,
:inl463:ind463,
:inl464:ind464,
:inl465:ind465,
:inl466:ind466,
:inl467:ind467,
:inl468:ind468,
:inl469:ind469,
:inl470:ind470,
:inl471:ind471,
:inl472:ind472,
:inl473:ind473,
:inl474:ind474,
:inl475:ind475,
:inl476:ind476,
:inl477:ind477,
:inl478:ind478,
:inl479:ind479,
:inl480:ind480,
:inl481:ind481,
:inl482:ind482,
:inl483:ind483,
:inl484:ind484,
:inl485:ind485,
:inl486:ind486,
:inl487:ind487,
:inl488:ind488,
:inl489:ind489,
:inl490:ind490,
:inl491:ind491,
:inl492:ind492,
:inl493:ind493,
:inl494:ind494,
:inl495:ind495,
:inl496:ind496,
:inl497:ind497,
:inl498:ind498,
:inl499:ind499,
:inl500:ind500,
:inl501:ind501,
:inl502:ind502,
:inl503:ind503,
:inl504:ind504,
:inl505:ind505,
:inl506:ind506,
:inl507:ind507,
:inl508:ind508,
:inl509:ind509,
:inl510:ind510,
:inl511:ind511,
:inl512:ind512,
:inl513:ind513,
:inl514:ind514,
:inl515:ind515,
:inl516:ind516,
:inl517:ind517,
:inl518:ind518,
:inl519:ind519,
:inl520:ind520,
:inl521:ind521,
:inl522:ind522,
:inl523:ind523,
:inl524:ind524,
:inl525:ind525,
:inl526:ind526,
:inl527:ind527,
:inl528:ind528,
:inl529:ind529,
:inl530:ind530,
:inl531:ind531,
:inl532:ind532,
:inl533:ind533,
:inl534:ind534,
:inl535:ind535,
:inl536:ind536,
:inl537:ind537,
:inl538:ind538,
:inl539:ind539,
:inl540:ind540,
:inl541:ind541,
:inl542:ind542,
:inl543:ind543,
:inl544:ind544,
:inl545:ind545,
:inl546:ind546,
:inl547:ind547,
:inl548:ind548,
:inl549:ind549,
:inl550:ind550,
:inl551:ind551,
:inl552:ind552,
:inl553:ind553,
:inl554:ind554,
:inl555:ind555,
:inl556:ind556,
:inl557:ind557,
:inl558:ind558,
:inl559:ind559,
:inl560:ind560,
:inl561:ind561,
:inl562:ind562,
:inl563:ind563,
:inl564:ind564,
:inl565:ind565,
:inl566:ind566,
:inl567:ind567,
:inl568:ind568,
:inl569:ind569,
:inl570:ind570,
:inl571:ind571,
:inl572:ind572,
:inl573:ind573,
:inl574:ind574,
:inl575:ind575,
:inl576:ind576,
:inl577:ind577,
:inl578:ind578,
:inl579:ind579,
:inl580:ind580,
:inl581:ind581,
:inl582:ind582,
:inl583:ind583,
:inl584:ind584,
:inl585:ind585,
:inl586:ind586,
:inl587:ind587,
:inl588:ind588,
:inl589:ind589,
:inl590:ind590,
:inl591:ind591,
:inl592:ind592,
:inl593:ind593,
:inl594:ind594,
:inl595:ind595,
:inl596:ind596,
:inl597:ind597,
:inl598:ind598,

```
:inl599:ind599,
:inl600:ind600,
:inl601:ind601,
:inl602:ind602,
:inl603:ind603,
:inl604:ind604,
:inl605:ind605,
:inl606:ind606,
:inl607:ind607,
:inl608:ind608,
:inl609:ind609,
:inl610:ind610,
:inl611:ind611,
:inl612:ind612,
:inl613:ind613,
:inl614:ind614,
:inl615:ind615,
:inl616:ind616,
:inl617:ind617,
:inl618:ind618,
:inl619:ind619,
:inl620:ind620,
:inl621:ind621,
:inl622:ind622,
:inl623:ind623,
:inl624:ind624,
:inl625:ind625,
:inl626:ind626,
:inl627:ind627,
:inl628:ind628,
:inl629:ind629,
:inl630:ind630,
:inl631:ind631,
:inl632:ind632,
:inl633:ind633,
:inl634:ind634,
:inl635:ind635,
:inl636:ind636,
:inl637:ind637,
:inl638:ind638,
:inl639:ind639,
:inl640:ind640,
:inl641:ind641,
:inl642:ind642,
:inl643:ind643,
:inl644:ind644,
:inl645:ind645,
:inl646:ind646,
:inl647:ind647,
:inl648:ind648,
:inl649:ind649,
:inl650:ind650,
:inl651:ind651,
:inl652:ind652,
:inl653:ind653,
:inl654:ind654,
:inl655:ind655,
:inl656:ind656,
:inl657:ind657,
:inl658:ind658,
:inl659:ind659,
:inl660:ind660,
:inl661:ind661,
:inl662:ind662,
:inl663:ind663,
:inl664:ind664,
:inl665:ind665,
:inl666:ind666,
:inl667:ind667,
:inl668:ind668,
:inl669:ind669,
:inl670:ind670,
:inl671:ind671,
:inl672:ind672,
:inl673:ind673,
:inl674:ind674,
:inl675:ind675,
:inl676:ind676,
:inl677:ind677,
:inl678:ind678,
:inl679:ind679,
:inl680:ind680,
:inl681:ind681,
:inl682:ind682,
:inl683:ind683,
:inl684:ind684,
:inl685:ind685,
:inl686:ind686,
:inl687:ind687,
:inl688:ind688,
:inl689:ind689,
:inl690:ind690,
:inl691:ind691,
:inl692:ind692,
:inl693:ind693,
:inl694:ind694,
:inl695:ind695,
:inl696:ind696,
:inl697:ind697,
:inl698:ind698,
:inl699:ind699,
:inl700:ind700,
:inl701:ind701,
:inl702:ind702,
:inl703:ind703,
:inl704:ind704,
:inl705:ind705,
:inl706:ind706,
:inl707:ind707,
:inl708:ind708,
:inl709:ind709 );

        if (sqlca.sqlcode < 0)
          sqlcode = error_check();    /* Don't bother printing the
SQL0100W
                                       error from going through the loop
                                       above one extra time */

        if (sqlcode == SQL_RC_E911)
        {       /* we've hit a deadlock */
#ifdef UF2DEBUG
  fprintf (myoutstream, "lineitem deadlocked\n");
  fflush(myoutstream);
#endif
          fprintf (myoutstream, "\nA deadlock has been detected for
LINEITEM ...Retrying...\n");
          SleepSome(UF_DEADLOCK_SLEEP);
          maxwait++;      /* jen DEADLOCK */
        }
        else if (sqlcode < 0)
        {
#ifdef UF2DEBUG
  fprintf (myoutstream, "lineitem failed\n");
  fflush(myoutstream);
#endif
          fprintf (myoutstream,"\nAn error occurred deleting from
TPCD.LINEITEM\n");
          fprintf (myoutstream,"for update pair number %d node %d chunk
%d..Exiting\n",
                updatePair, i,deleteChunk);
          rc=-1;
        }
        else
        {
```

```
#ifdef UF2DEBUG
  fprintf (myoutstream, "lineitem succeeded\n");
  fflush(myoutstream);
#endif
            /* report the number of row deleted */
            fprintf(myoutstream, "%ld rows deleted from TPCD.LINEITEM
at %18.18s\n",
                    sqlca.sqlerrd[2],
                    get_time_stamp(T_STAMP_FORM_1,(time_t)NULL));

            rc=0;
            EXEC SQL COMMIT WORK;
            error_check();
          }
        } /* process lineitem deletes */
      } /* while trying to delete one chunk loop */
    } /* while processing keys to delete */

#ifdef UF2DEBUG
  fprintf (myoutstream, "after loop\n");
  fflush(myoutstream);
#endif

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

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

UF2_exit:
  fclose (myoutstream);
  /* exiting, increment the semaphore */
#ifdef TPCD_NONPARTITIONED
#ifndef SQLWINT
  /* we used the first flat file to generate the semaphore key */
  if (getenv("TPCD_FLATFILES") != NULL)
    sprintf(sourcefile, "%s%cdelete.%d.0",
          getenv("TPCD_FLATFILES"), PATH_DELIM, updatePair);
  else
    sprintf(sourcefile,".%cdelete.%d.0", PATH_DELIM,
          updatePair);
  su_semkey = ftok (sourcefile, 'J');
  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",
            i, errno);
      exit(1);
    }
  }
  if (sem_op (su_semid, 0, 1) != 0 )     /*jenSEM*/
  {                                 /*jenSEM*/
    fprintf(stderr,"Failure to increment semaphone UF2 set %d\n",i);
    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
#endif /* TPCD_NONPARTITIONED */
  exit(rc);                 /* child exiting after finishing up */
}

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

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

/*************************************************************
******/
/* Determines the proper name for the output file to
  be generated for a particular TPC-D query, update function, or
  interval summary                                          */
/*************************************************************
******/
void output_file(struct global_struct *g_struct)
{
  char file_name[256] = "\0";
  char run_dir[150]   = "\0";
  char time_stamp[50] = "\0";
  char delim[2]       = "\0";
  int qnum;


  strcpy(run_dir,g_struct->run_dir);
  sprintf(delim,"%s",getenv("TPCD_PATH_DELIM"));
  strcpy(time_stamp,g_struct->file_time_stamp);

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

```
    }

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

  switch (g_struct->c_l_opt->intStreamNum)
  {
   case -1: /* qualifiying */
     sprintf(file_name,
"%s%sqryqual%02d.%s",run_dir,delim,qnum,time_stamp);
     break;
   case 0: /* power tests */
     if (qnum < 0) /* update functions */
       sprintf(file_name,
"%s%smpuf%d.%02d.%s",run_dir,delim,abs(qnum), \
               currentUpdatePair,time_stamp);
     else
       sprintf(file_name,
"%s%smpqry%02d.%s",run_dir,delim,qnum,time_stamp);
     break;

   default:
     /*      if (qnum < 0)  - replaced by berni 96/03/26 */
     if (g_struct->c_l_opt->update == 2 ||
         g_struct->c_l_opt->update == 5)
       sprintf(file_name,
"%s%smtuf%d.%02d.%s",run_dir,delim,abs(qnum), \
               currentUpdatePair,time_stamp);
     else
       sprintf(file_name, "%s%smts%dqry%02d.%s",run_dir,delim, \
               g_struct->c_l_opt->intStreamNum,qnum,time_stamp);
     break;

  }

  if (g_struct->c_flags->eo_infile)
    if (g_struct->c_l_opt->update == 2 ||
        g_struct->c_l_opt->update == 5)
      sprintf(file_name, "%s%smtufinter.%s",run_dir,delim,time_stamp);
    else
      switch (g_struct->c_l_opt->intStreamNum) {
       case -1:
         sprintf(file_name,
"%s%sqryqualinter.%s",run_dir,delim,time_stamp);
         break;
       case 0:
         sprintf(file_name, "%s%smpinter.%s",run_dir,delim,time_stamp);
         break;
       default:
         if (g_struct->c_l_opt->intStreamNum > 0)
           sprintf(file_name,
                 "%s%smts%dinter.%s",
                 run_dir,delim,g_struct->c_l_opt-
>intStreamNum,time_stamp);
         else
           fprintf(stderr,"Invalid stream number specified\n");
         break;
      }

  strcpy(outstreamfilename, file_name); /* wlc 081397 */

  if (!feof(instream) || g_struct->c_flags->eo_infile)
    /* Only create an output file if there are input
       statements left to process, or if we're all done
       and want to print out the summary table file */
    if( (outstream = fopen(file_name, WRITEMODE)) == NULL ) {
      fprintf(stderr,"\nThe output file could not be opened.  ");
      fprintf(stderr,"Make sure that the filename is correct.\n");
      fprintf(stderr,"filename = %s\n",file_name);
      exit(-1);
    }

  return;
}


/*************************************************************
******/
/* Determine whether or not we should break out of the block loop
   because of an end of file, end of block, or update function.
   Also handle some semaphore stuff for update functions        */
/*************************************************************
******/
int PreSQLprocess(struct global_struct *g_struct)
{
  int             rc = 1;
  FILE              *updateFP;
#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

  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 ( ( g_struct->sem_on == 1 ) &&
          ( g_struct->c_l_opt->intStreamNum != 0 ) )
     {
       /* yes we are to be using semaphores */
       /* is this the 1st time into update function 1??? */
       if (currentUpdatePair == updatePairStart )
       {
#ifdef SQLWINT
         fprintf(stderr,"numstreams = %d\n",g_struct->c_l_opt-
>intStreamNum);
         fprintf(stderr,"semfile = %s\n",g_struct->sem_file);
         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");
          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,"Done waiting! 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
          /* create a semaphore key...use the name of a file that */
          /* you know exists */

          semkey = ftok(g_struct->update_num_file,'J');
          if ( (semid = semget(semkey,1,IPC_CREAT|S_IRUSR|S_IWUSR))
< 0)
          {
            fprintf(stderr,
                  "Throughput can't get initial semaphore! semget failed errno
= %d\n",
                  errno);
            exit(1);
          }
          if (verbose)
          {
            fprintf(stderr,
                  "insert: semkey = %ld, semid = %d, file = %s, value =
%d\n",
                  semkey,semid,g_struct->update_num_file,
                  (g_struct->c_l_opt->intStreamNum * -1));
          }

          /* 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\n");
          semctl(semid,0,IPC_RMID,0);  /* we've finished waiting, now */
                                  /* remove the semaphore */
#endif
        }
        /* otherwise continue to run*/
      }

      if (g_struct->c_l_opt->update != 5) {
        runUF1(currentUpdatePair, g_struct->copy_on_load);
      }
      rc = FALSE;
      break;
     case TPCDBATCH_DELETE:
      if (g_struct->c_l_opt->update != 5) {
        runUF2(currentUpdatePair, g_struct->copy_on_load);
      }
      currentUpdatePair += 1;
      /* update the update.pair.num file to reflect the successfully completed
*/
      /* update pair */
      if (g_struct->c_l_opt->update != 5)
      {    /*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;
   int rows_fetch = 0;
   long sqlcode = SQL_RC_E911;             /* Temporary sqlcode to test
                                           for deadlocks */
   int max_wait = 1;                       /* Maximum number of retries
                                           for deadlock scenario */

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

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

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



   while ( ( (sqlcode == SQL_RC_E911) ||
           (sqlcode == SQL_RC_E912) ) && (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. Aborting this query.\n");
        rc = -1;
      }
      else
      {
        /* print out the column headings for the answer set */
        print_headings(sqlda,col_lengths);              /* @d22817 tjg */

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

        EXEC SQL DECLARE DYNCUR CURSOR FOR STMT1;

        EXEC SQL OPEN DYNCUR;
        sqlcode = error_check();

        if (sqlcode == SQL_RC_E912)           /* we ran out of locks */
        {
          max_wait ++;
          fprintf (stderr, "\nA -912 (out of locks) error has been detected on
open...Retrying...\n");
          SleepSome(10);
        }
        else
        {

/*************************************************************
***********/
          /* Fetch appropriate number of rows and determine whether or
not to    */
          /* send them to file.                                  */

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

          rows_fetch = 0;

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

            EXEC SQL FETCH DYNCUR USING DESCRIPTOR :*sqlda;
            if (sqlca.sqlcode == 100)
            {
              sqlcode = sqlca.sqlcode;
            }
            else
            {
              sqlcode = error_check();
            }
            if (sqlcode == 0)
            {
              rows_fetch++;
              if ( (rows_fetch <= s_info_ptr->max_rows_out) ||
                   (s_info_ptr->max_rows_out == -1) )
                echo_sqlda(sqlda,col_lengths);
            }
            else if (sqlcode == SQL_RC_E912)
            {
              max_wait ++;
              fprintf (stderr, "\nA -912 (out of locks) error has been
detected on fetch...Retrying...\n");
              SleepSome(10);
            }
          }  while ( (sqlcode == 0) && \
                     ( (s_info_ptr->max_rows_fetch == -1) || \
                       (rows_fetch < s_info_ptr->max_rows_fetch) ) );
        } /*   end of successful open */
      } /* end of successful prepare */
    } /** End of block for handling SELECT statements **/
    else
    {       /** SQL statement is not a SELECT **/
      EXEC SQL EXECUTE IMMEDIATE :stmt_str;
      sqlcode = error_check();

      if (sqlcode == SQL_RC_E911)         /* we've hit a deadlock */
      {
        max_wait ++;
        fprintf (stderr, "\nA deadlock has been detected on execute
immediate...Retrying...\n");
        SleepSome(10);
      }
    } /* end of block for handling NON-select statements */

    if  ( (sqlcode >=0 ) &&
        (g_struct->c_flags->select_status == TPCDBATCH_SELECT) )
    {
      /* we opened a cursor before */
      EXEC SQL CLOSE DYNCUR;
      sqlcode = error_check();

      if ((s_info_ptr->max_rows_fetch == -1) || \
         (rows_fetch < s_info_ptr->max_rows_fetch))
        fprintf (outstream, "\n\nNumber of rows retrieved is: %6 %d", \
              rows_fetch);
      else
        fprintf (outstream, "\n\nNumber of rows retrieved is: %6 %d", \
              s_info_ptr->max_rows_fetch);
    }                                       /* @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 function 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;

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

  s_info_ptr = g_struct->s_info_ptr;


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

  if (g_struct->c_l_opt->a_commit) {
    EXEC SQL COMMIT WORK;
    error_check();                          /* @d22275 tjg */
  }

  s_info_ptr->elapse_time = get_elapsed_time(start_time);
```

```
  if (g_struct->c_flags->time_stamp == TRUE)              /* @d25594 tjg
*/
    strcpy(s_info_ptr->end_stamp,
    get_time_stamp(T_STAMP_FORM_1,(time_t)NULL) );

  /* write the start timestamp to the file */
  fprintf( outstream,"\n\nStop timestamp %18.18s \n",
         s_info_ptr->end_stamp);



  fflush(outstream);

  /** Allocate space for a new stmt_info structure **/    /* @d24993 tjg */
  s_info_ptr->next =
    (struct stmt_info *) malloc(sizeof(struct stmt_info));
  if (s_info_ptr->next != NULL)  {
    memset(s_info_ptr->next, '\0', sizeof(struct stmt_info));
    /** Transfer details from one structure to another for
      to apply for the next statement **/
    s_info_ptr->next->stmt_num = s_info_ptr->stmt_num + 1;
    s_info_ptr->next->max_rows_fetch = s_info_ptr->max_rows_fetch;
    s_info_ptr->next->max_rows_out = s_info_ptr->max_rows_out;

    s_info_ptr->next->query_block = s_info_ptr->query_block;
    s_info_ptr->next->elapse_time = -1;

    s_info_ptr = s_info_ptr->next;

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

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

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

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

  return (TRUE);
}

/*****************************************************************
*******************/
/* Does some cleaning up once all the statements are processed.
Disconnects
  from the database, cleans up some semaphore stuff from the update
functions,
  prints out the summary table, and closes all file handles.              */
/*****************************************************************
*******************/
int cleanup(struct global_struct *g_struct)
{
#ifndef SQLWINT
  int             semid;          /* semaphore for controlling UFs*/
  key_t           semkey;         /* key to generate semid */
#else
  HANDLE          hSem;
#endif

  /** End timestamp for stream **/
  g_struct->stream_end_time = time(NULL);

  if (g_struct->c_l_opt->intStreamNum > 0)
  {
    /* print out the stream stop time in the stream count information file*/
    /* for the throughput tests only (intStreamNum>0)*/
    if (g_struct->c_l_opt->update == 2 ||
       g_struct->c_l_opt->update == 5)
    {
      /* update function stream */
      fprintf(g_struct->stream_report_file,
            "Update function stream stopping at %18.18s\n",
            get_time_stamp(T_STAMP_FORM_1,g_struct-
>stream_end_time));
    }
    else
    {
      /* query streams */
      fprintf(g_struct->stream_report_file,
            "Stream number %d stopping at %18.18s\n",
            g_struct->c_l_opt->intStreamNum,
            get_time_stamp(T_STAMP_FORM_1,g_struct-
>stream_end_time));
    }
    fclose(g_struct->stream_report_file);
  }

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

  /* 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 ( ( g_struct->sem_on == 1 ) &&
      ( g_struct->c_l_opt->update == 0) &&
      ( g_struct->c_l_opt->intStreamNum > 0 ) )
  {



    /* create a semaphore key...use the name of a file that */
    /* you know exists */

#ifndef SQLWINT

    semkey = ftok(g_struct->update_num_file,'J');
    /* Okay, now's the time to increment the semaphore by the update
stream */
    while ((semid = semget(semkey,1,0)) < 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);
      }
    }
    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);
    }
    /* this stream is done...increment by one */
    if (sem_op(semid, 0, 1) != 0)                   /*jenSEM*/
```

```
    {                                       /*jenSEM*/
      fprintf(stderr,
           "Failed to increment semaphone for throughput stream %d\n",
           g_struct->c_l_opt->intStreamNum);
      fprintf(stderr,
           "file for generation of semaphore is: %s\n",
           g_struct->update_num_file);
      exit(1);
    }                                       /*jenSEM*/


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

      Sleep(1000);
    }

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

D.2 Makefile.pe

```
#############################################################
##########
# MAKEFILE for tpcdbatch program
# Enter the Following:
#
#     make tpcdbatch     -- makes tpcdbatch
#
#     make cleanup       -- removes builds from tpcdbatch program
#
#############################################################
##########

#LOCAL=tpcd

BASE=$(HOME)/sqllib
LINK_FLAGS= -o $@ -L/usr/lpp/db2_05_00/lib -ldb2
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
2>/dev
/null

all : tpcdbatch

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

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

D.3 load_update

```
#!/bin/ksh

if [ $# -lt 2 ]; then
  echo "Usage: $0 pair function <inlistmax>"
  exit -1;
fi;

updatepair=$1
function=$2
inlistmax="";
if [ $# -eq 3 ]; then
  inlistmax=$3;
fi
HOME=/u/tpcd;

dsh -a ". /u/tpcd/.profile; /u/tpcd/tpcd/tools/load_update_on_node
$updatepair $
function $inlistmax"

exit 0;
```

D.4 load_update_on_node

```
#!/bin/ksh

if [ $# -lt 2 ]; then
  echo "Usage: $0 pair function <inlistmax>"
  exit -1;
fi;

updatepair=$1
function=$2
inlistmax="";
if [ $# -eq 3 ]; then
  inlistmax=$3;
fi
HOME=/u/tpcd;

function get_envs {
  envdir=$PWD;
  sed 's/\#.*$//g' $envdir/tpcd.setup > /tmp/tpcd.envs
  sed 's/\#.*$//g' $envdir/tpcd.runsetup >> /tmp/tpcd.envs
  sed '/^ *$/d' /tmp/tpcd.envs > /tmp/tpcd.envs2
  sed 's/^/export /' /tmp/tpcd.envs2 > /tmp/tpcd.envs
  . /tmp/tpcd.envs
#  rm /tmp/tpcd.envs /tmp/tpcd.envs2
}

function runInsert_concat {
#set -x;
     db2 connect to $dbname;
     str="db2 \"load from ";
    str2="db2 \"load from ";
```

```
str="$str$flatfilepath${delim}lineitem.tbl.u${updatepair}.${LN3}.new";

str2="$str2$flatfilepath${delim}order.tbl.u${updatepair}.${LN3}.new";
  str="$str of del modified by coldel| fastparse messages
${tempdir}${delim}lin
e.msg.u${updatepair}.${LN3}.new remote file /tmp/tpcd/remote.uf1_line
replace in
to TPCDTEMP.LINEITEM_new nonrecoverable \" >
${tempdir}${delim}loaduf${function}
.lineitem.u${updatepair}.${LN3} 2>&1";
  str2="$str2 of del modified by coldel| fastparse messages
${tempdir}${delim}o
rd.msg.u${updatepair}.${LN3}.new remote file /tmp/tpcd/remote.uf1_ord
replace in
to TPCDTEMP.ORDERS_new nonrecoverable \" >
${tempdir}${delim}loaduf${function}.o
rder.u${updatepair}.${LN3} 2>&1";

#    print -- "$str";
#    print -- "$str2";
   eval "$str";
   eval "$str2";
 db2 connect reset; db2 terminate;
 return;
}

function runDelete {
 typeset -i chk=0;
 typeset -i ln;
 (( ln=LN3 ))

 while [ chk -lt split_deletes ]; do
#    jj=0;
#    while [ jj -lt 48 ]; do
     $auditDir${delim}auditruns${delim}tpcdbatch -z -d $dbname -i
$updatepair
-j 2 -k $ln -m $inlistmax -x $chk > $tempdir${delim}uf2.$ln.$chk.out
2>&1 &
#     (( jj=jj+1 ));
    (( chk=chk+1 ))
#   done
#   wait;
 done;
 wait;
}

if [ $# -lt 2 ]; then
 echo "Usage: $0 pair function"
 exit -1;
fi;

cd $HOME/tpcd/tools;

updatepair=$1
function=$2
if [ $# -eq 3 ]; then
 inlistmax=$3;
fi;
#get_envs;
. /tmp/tpcd.envs;

export TPCD_PATH_DELIM="/";

hnm=$(hostname -s);
typeset -Z5 LN3;
LN3=$(grep $hnm $HOME/sqllib/db2nodes.cfg.ethernet | head -1 | cut -f1
-d" ");
dbname=$TPCD_DBNAME;
auditDir=$TPCD_AUDIT_DIR;
uftemp=$TPCD_UFTEMP;
delim=$TPCD_PATH_DELIM;
#split_updates=$TPCD_SPLIT_UPDATES;
split_updates=48;
concurrentload=$TPCD_CONCURRENT_INSERTS_LOAD;
split_deletes=$TPCD_SPLIT_DELETES;
uftemppath=$TPCD_UFTEMPPATH;
platform=$TPCD_PLATFORM;
flatfilepath=$TPCD_FLATFILES;
tempdir=$TPCD_TMP_DIR;
physnode=$TPCD_PHYS_NODE;
lnperpn=$TPCD_LN_PER_PN;

if [ function -eq 1 ]; then
runInsert_concat;
else if [ function -eq 2 ]; then
  runDelete;
else
  echo "Problem in function number"
  exit -1;
fi
fi
```

D.5 runpower

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

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

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

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

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

# 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";
}

#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"};

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

# the auditruns directory is where we have already generate the sql files for
th
e
# 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");
}

# activate the database
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 : $cu
rTs\n";
close(MISO);
if ( $product eq "pe" )
{
   system("db2 \"connect to $dbname\"; db2 \"select
name,creator,valid,unique_i
d,isolation from sysibm.sysplan where name='TPCDBATC'\"; 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 tes
ts
  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\" >>
$runDir/s
ysunused$runNum");
  $curTs = `perl gettimestamp "long"`;
  system("echo \"$curTs\" >> $runDir/sysunused$runNum");
  system("lsvg -o >> $runDir/sysunused$runNum");
  # show the disks that are used/unused
  system("getdisks \"Before the start of the Power Test\"");

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

&getConfig("p");
if ($gatherstats eq "on")
{
   # gather vm io and net stats
   if ($platform eq "aix")
   {
     # 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";
   }
}
if ( $runUF ne "UF" )
{
  print "Beginning power stream....no update functions\n";
  # run the 17 queries for the powertest (stream = 0) with NO update
functions
  $ret=system("$auditDir${delim}auditruns${delim}tpcdbatch -d $dbname
-f $runDir
```

```
${delim}qtextpow.sql -r on -b on -s $sf -u p -m $inlistmax -n 0 -l
$auditDir${de
lim}auditruns${delim}querytext${delim}streampow.list");
}
else
{
  print "Beginning power stream....with update functions\n";
  # run the 17 queries for the powertest (stream = 0) with the update
functions
  $ret=system("$auditDir${delim}auditruns${delim}tpcdbatch -d $dbname
-f $runDir
${delim}qtextp.sql -r on -b on -s $sf -u p -m $inlistmax -n 0 -l
$auditDir${deli
m}auditruns${delim}querytext${delim}stream0.list");
}
if ($ret == 0)
{
   print "Power stream completed succesfully.\n";
}
else
{
   print "Power stream failed. ret=$ret\n";
}

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

   #clean up
}
if ($gatherstats eq "on")
{
   # gather vm io and net stats
   if ($platform eq "aix")
   {
     # kill the stats that were being gathered
      $rc= `perl5 zap "-f" "vmstat"`;
      $rc= `perl5 zap "-f" "iostat"`;
      if ( $pn > 1 )
      {
         $rc= `perl5 zap "-f" "netstat"`;
      }
      $rc= `perl5 zap "-f" "getstats"`;
   }
}


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

if ( $product eq "pe" )
{
   system("db2 \"connect to $dbname\"; db2 \"select
name,creator,valid,unique_i
d,isolation from sysibm.sysplan where name='TPCDBATC'\";db2 connect
reset;db2 te
rminate  >> $runDir${delim}miso$runNum");
}
else
{
   &verifyTPCDbatch("$misofile","$dbname");
}

# deactivate the database
system("db2 deactivate database $dbname");

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 :
$timestam
p";
  close(DBTUNE);
  system("db2 get database configuration for $dbname >> $dbtunefile");
  system("db2 get database manager configuration >> $dbtunefile");
  system("db2set >> $dbtunefile");

}

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.s
ysplan where name='TPCDBATC';\n";
  print VERTBL "connect reset;\n";
  print VERTBL "terminate;\n";
  close(VERTBL);
  system("db2 -vtf $file >> $logfile");
}
```

D.6 runthroughput

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

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

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

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

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

$runUF="no";
if (@ARGV > 0)
{
  if ($ARGV[0] eq "UF")
  {
    $runUF=$ARGV[0];
    if ( @ARGV > 1)
    {
       $numStream=$ARGV[1];
```

```
    }
    else
    {
       $numStream=0;
    }
    shift;
  }
  else
  {
     $numStream=$ARGV[0];
  }
}
else
{
    $numStream=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
";
}

#set up local variables
$runNum=$ENV{"TPCD_RUNNUMBER"};
if ( $numStream == 0 )
{
  $numStream=$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"};

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

# if we are in real audit mode then we must start the db manager now
if ( $RealAudit eq "yes" )
{
    system("db2start");
}

# activate the database
system("db2 activate database $dbname");

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

if ( $product eq "pe" )
{
    system("db2 \"connect to $dbname\"; db2 \"select
name,creator,valid,unique_i
d,isolation from sysibm.sysplan where name='TPCDBATC'\" >>
$runDir${delim}miso$r
unNum ");
}
else
{
    &verifyTPCDbatch("$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.
if ( $platform eq "aix" )
{
   system("watchstreams &");
}
elsif ( $platform eq "nt" )
{
    system("start perl watchstreams");
}
else
{
    die "platform not supported, can't start watchstreams in background";
}
```

```
# show the disks that are used/unused
if ($platform eq "aix")
{
   system("getdisks \"Before the start of the Throughput Test\"");
}
if ($gatherstats eq "on")
{
   # gather vm io and net stats
   if ($platform eq "aix")
   {
     # 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 17 queries
  if ( $platform eq "aix" )
  {
     system("$auditDir${delim}auditruns${delim}tpcdbatch -d $dbname -
f $runDi
r${delim}qtextt$loopStream.sql -r on -b on -s $sf -u t1 -m $inlistmax -n
$loopSt
ream -l
$auditDir${delim}auditruns${delim}querytext${delim}stream$loopStrea
m.lis
t &");
  }
  elsif ( $platform eq "nt" )
  {
     system("start /b $auditDir${delim}auditruns${delim}tpcdbatch -d
$dbname
-f $runDir${delim}qtextt$loopStream.sql -r on -b on -s $sf -u t1 -m
$inlistmax -
n $loopStream -l
$auditDir${delim}auditruns${delim}querytext${delim}stream$loopS
tream.list");
  }
  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 $audi
tDir${delim}auditruns${delim}quft.sql -r on -b on -s $sf -u t -m
$inlistmax -n $
numStream -l
$auditDir${delim}auditruns${delim}querytext${delim}streamuf.list");

 }
 else {
   $ret=system("$auditDir${delim}auditruns${delim}tpcdbatch -d
$dbname -f $audi
tDir${delim}auditruns${delim}quft.sql -r on -b on -s $sf -u t2 -m
$inlistmax -n
$numStream -l
$auditDir${delim}auditruns${delim}querytext${delim}streamuf.list")
;
 }
 print "update stream done\n";


 &getConfig("t");
}
else
{
 # we are running the query streams and update stream from different
nodes, use

 # files and rksh to control the update stream
 system("runthru.pe");
}

if ($platform eq "aix")
{
   # show the disks that are used/unused
   system("getdisks \"After the completion of the Throughput Test\"");
}
if ($gatherstats eq "on")
{
   # gather vm io and net stats
   if ($platform eq "aix")
   {
     # kill the stats that were being gathered
      $rc= ‘perl5 zap "-f" "vmstat"‘;
      $rc= ‘perl5 zap "-f" "iostat"‘;
      if ( $pn > 1 )
      {
         $rc= ‘perl5 zap "-f" "netstat"‘;
      }
      $rc= ‘perl5 zap "-f" "getstats"‘;
   }
}

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

if ( $product eq "pe" )
{
   system("db2 \"connect to $dbname\"; db2 \"select
name,creator,valid,unique_i
d,isolation from sysibm.sysplan where name='TPCDBATC\" >>
$runDir${delim}miso$r
unNum");
}
else
{
  &verifyTPCDbatch("$misofile","$dbname");
}

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

```
# calcmetrics.pl, and because the output from calcmetrics is used as
# the trigger for watchstreams to complete, and watchstreams cats its
# output at the end of the mstrcnt file.

# generate the mtinter?.metrics file in the run directory
#require 'calcmetrics.pl';
if ( $platform eq "aix" )
{
  system("calcmetrics.pl $numStream");
}
elsif ( $platform eq "nt" )
{
   system("perl calcmetrics.pl $numStream");
}
else
{
   die "platform not supported, can't run calcmetrics ";
}

# concatenate all the throughput inter files that were used to
# generate these results into the calcmetrics output file (mtinterX.metrics)
#cd $TPCD_RUN_DIR
&cat("$runDir${delim}mts*inter*","$runDir${delim}mtinter$runNum.me
trics");

if ($runUF ne "no") {

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

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

  # generate the mlog information file
  require 'buildmlog';
}

if ($runUF eq "no") {
  &rm("$runDir${delim}mtuf*");
}

# deactivate the database
system("db2 deactivate database $dbname");

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 :
$timestam
p";
  close(DBTUNE);
  system("db2 get database configuration for $dbname >> $dbtunefile");
  system("db2 get database manager configuration >> $dbtunefile");
  system("db2set >> $dbtunefile");

}

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.s
ysplan where name='TPCDBATC';\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

```
/*************************************************************
************/
/*      File: acid.sqc                                       */
/*************************************************************
************/

/*   changes:
 *
 *  961109  jel    add EXEC SQL CLOSE for each cursor in acidT
 *                 to avoid bug in db2pe v1r2
 *
 */

#include "acid.h"

#define DEADLOCK -911

#define TRUNC2(d) ((floor((d)*100))/100)

void sqlerror(char * , struct sqlca *);

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

/*-------------------------------------------------------------*/
/*       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;
  long    okey;
  long    lEprice;
  double  eprice;
  EXEC SQL END DECLARE SECTION;

  okey = acid->o_key;

  /* mypid = getpid(); */
  mypid = acid->tag;
  sprintf(out_fn, "/tmp/tpcd/acidQ.out.%d",mypid);
  out=fopen(out_fn,"a");
  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));

  EXEC SQL SELECT
SUM(DECIMAL(L_EXTENDEDPRICE,20,2)*100) into :lEprice
    from TPCD.LINEITEM where L_ORDERKEY = bigint(: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 = (float)lEprice / 100.00;  /* translate to float 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): %0.2f\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);
  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;
  long int_cost;
  long lfint_cost;
  float fint_cost;

  EXEC SQL BEGIN DECLARE SECTION;
  long      o_key, l_key, delta;
  long      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 = bigint(: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 = bigint(: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, "/tmp/tpcd/acidT.out.%d",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;
     fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
     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;
     fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
     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 l_orderkey = bigint(:o_key)
     AND l_linenumber = :l_key;
/*   WHERE CURRENT OF l_cursor; */
   if (sqlca.sqlcode != 0) {
     if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
     rc = sqlca.sqlcode;
     fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
     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;
       fprintf(out,"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;
```

```
    fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
    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;
    fprintf(out,"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);
  }

  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("ototal= %0.3f\n",ototal);
  printf("new_ototal= %0.3f\n",new_ototal);
#endif

  EXEC SQL UPDATE tpcd.orders
    SET o_totalprice = :new_ototal
    WHERE o_orderkey = bigint(:o_key);
/*   WHERE CURRENT OF o_cursor; */
  if (sqlca.sqlcode != 0) {
    if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
    rc = sqlca.sqlcode;
    fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
    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) ;
  }

  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;
      fprintf(out,"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;
    fprintf(out,"acidT **ERROR** sqlcode = %d\n",sqlca.sqlcode);
    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 */
  if ( ((random() % (100)) + 1) < 80 ) sleep(1);

  switch (acid->termination) {
   case 1:
    {
     if (acid->logging)
     {
      gettimeofday(&tv, &tz);
      time(&timeT);
      fprintf(out,"acidT tag: %d, wait before COMMIT: (%us %06uu)
%s",
            mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
     }
    }
    sleep(60);
   case 0:
    if (acid->logging) {
      gettimeofday(&tv, &tz);
      time(&timeT);
      fprintf(out,"acidT tag: %d, immediately before COMMIT: (%us
%06uu) %s",
            mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    }
    EXEC SQL CLOSE L_CURSOR;
    EXEC SQL CLOSE O_CURSOR;
    EXEC SQL COMMIT;
    if (sqlca.sqlcode != 0) {
      if(sqlca.sqlcode == DEADLOCK) goto retry_tran;
      rc = sqlca.sqlcode;
      fprintf(out,"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, after COMMIT: (%us %06uu) %s",
            mypid, tv.tv_sec, tv.tv_usec, ctime(&timeT));
    }
    break;
   case 3:
    if (acid->logging) {
      gettimeofday(&tv, &tz);
      time(&timeT);
```

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

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

  rc = 0;
  goto Texit;

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

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

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

  qnum = us->qnum;

  /*sprintf(out_fn, "/tmp/tpcd/update.out.%d",qnum); */
  out=fopen("/tmp/tpcd/update.out","a");
  gettimeofday(&tv, &tz);
  time(&timeT);
  fprintf(out,"\n---------- START of update ----------\n\n");
  fprintf(out, "update query number: %d, begin transaction time: (%us
%06uu) %s",
        qnum, tv.tv_sec, tv.tv_usec, ctime(&timeT));

  sqlca.sqlcode = 0;
  discount = 0.25;
  price = 5000.50;
  acctbal = 1000.00;
  availqty = 10;
  size = 5;

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

    switch (qnum)
    {
     case 1:
     {
       EXEC SQL
         UPDATE TPCD.LINEITEM set L_DISCOUNT = L_DISCOUNT
+ :discount
         WHERE L_ORDERKEY IN
(bigint(326),bigint(512),bigint(928),bigint(995));
       if (sqlca.sqlcode != 0) {
         rc = sqlca.sqlcode;
         fprintf(out,"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;
         fprintf(out,"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 (bigint(260930), bigint(402497),
bigint(457859), bigint(509889), bigint(58117),
                        bigint(538311), bigint(588421), bigint(416167),
bigint(97830), bigint(90276));
        if (sqlca.sqlcode != 0) {
          rc = sqlca.sqlcode;
          fprintf(out,"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 =  bigint(67461);
        } else {
          EXEC SQL
            UPDATE TPCD.ORDERS set O_ORDERDATE =
O_ORDERDATE + 6 MONTHS
            WHERE O_ORDERKEY =  bigint(67461);
        }
        if (sqlca.sqlcode != 0) {
          rc = sqlca.sqlcode;
          fprintf(out,"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
(bigint(70976),bigint(566279),bigint(152897),bigint(84226),bigint(23248
3));
        if (sqlca.sqlcode != 0) {
          rc = sqlca.sqlcode;
          fprintf(out,"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
(bigint(33),bigint(131),bigint(161),bigint(195),bigint(229),bigint(230),bigi
nt(231),bigint(323),bigint(353),bigint(356));
        if (sqlca.sqlcode != 0) {
          rc = sqlca.sqlcode;
          fprintf(out,"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
(bigint(562917),bigint(410659),bigint(16550),bigint(398401),bigint(1576
34),bigint(429920),bigint(45411));
        if (sqlca.sqlcode != 0) {
          rc = sqlca.sqlcode;
          fprintf(out,"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
(bigint(129569),bigint(343591),bigint(270242),bigint(254983),bigint(985
00),bigint(28963));
        if (sqlca.sqlcode != 0) {
          rc = sqlca.sqlcode;
          fprintf(out,"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
(bigint(113509),bigint(232997),bigint(246691),bigint(379233),bigint(448
162),bigint(32134));
        if (sqlca.sqlcode != 0) {
          rc = sqlca.sqlcode;
          fprintf(out,"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
(bigint(516487),bigint(245411),bigint(265799),bigint(253025),bigint(691
4),bigint(562020));
      if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        fprintf(out,"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;
        fprintf(out,"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
(bigint(33),bigint(70),bigint(195),bigint(355),bigint(677),bigint(837),bigin
t(960),bigint(962),bigint(1028));
      } else {
        EXEC SQL
          UPDATE TPCD.LINEITEM set L_RECEIPTDATE =
L_RECEIPTDATE + 3 YEARS
          WHERE L_ORDERKEY IN
(bigint(33),bigint(70),bigint(195),bigint(355),bigint(677),bigint(837),bigin
t(960),bigint(962),bigint(1028));
      }
      if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        fprintf(out,"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
(bigint(263),bigint(9476),bigint(32355),bigint(34854),bigint(53445),bigint
(56901));
      if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        fprintf(out,"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
(bigint(32),bigint(225),bigint(326),bigint(448),bigint(449),bigint(483),bigi
nt(512));
      if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        fprintf(out,"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
(bigint(1),bigint(4),bigint(7),bigint(35),bigint(135),bigint(131300));
      if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        fprintf(out,"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;
        fprintf(out,"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
(bigint(4065),bigint(110372),bigint(165061),bigint(265702),bigint(87138)
);
        if (sqlca.sqlcode != 0) {
          rc = sqlca.sqlcode;
          fprintf(out,"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);
    fprintf(out,"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);
    fprintf(out,"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");
  fclose(out);
  return(rc);
}

/*-------------------------------------------------------------*/
/*      connect_to_TM                                          */
/*-------------------------------------------------------------*/
void connect_to_TM( void )
{
  char *dbname_ptr;
  if ((dbname_ptr = getenv("TPCD_QUAL_DBNAME")) != NULL) {
    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[] = "/tmp/tpcd/acid.err.out";
  int j,k;

  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");
  fclose(err_fp);

}
```