

TPC Benchmark™ H Full Disclosure Report
for
IBM® @server™ 325
using
IBM DB2® Universal Database 8.1

Submitted for Review

July 29, 2003



Third Edition – December 2005

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

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

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

© Copyright International Business Machines Corporation 2003. All rights reserved.

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

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

Trademarks


IBM, the IBM e-business logo, DB2 are trademarks or registered trademarks of International Business Machines Corporation.

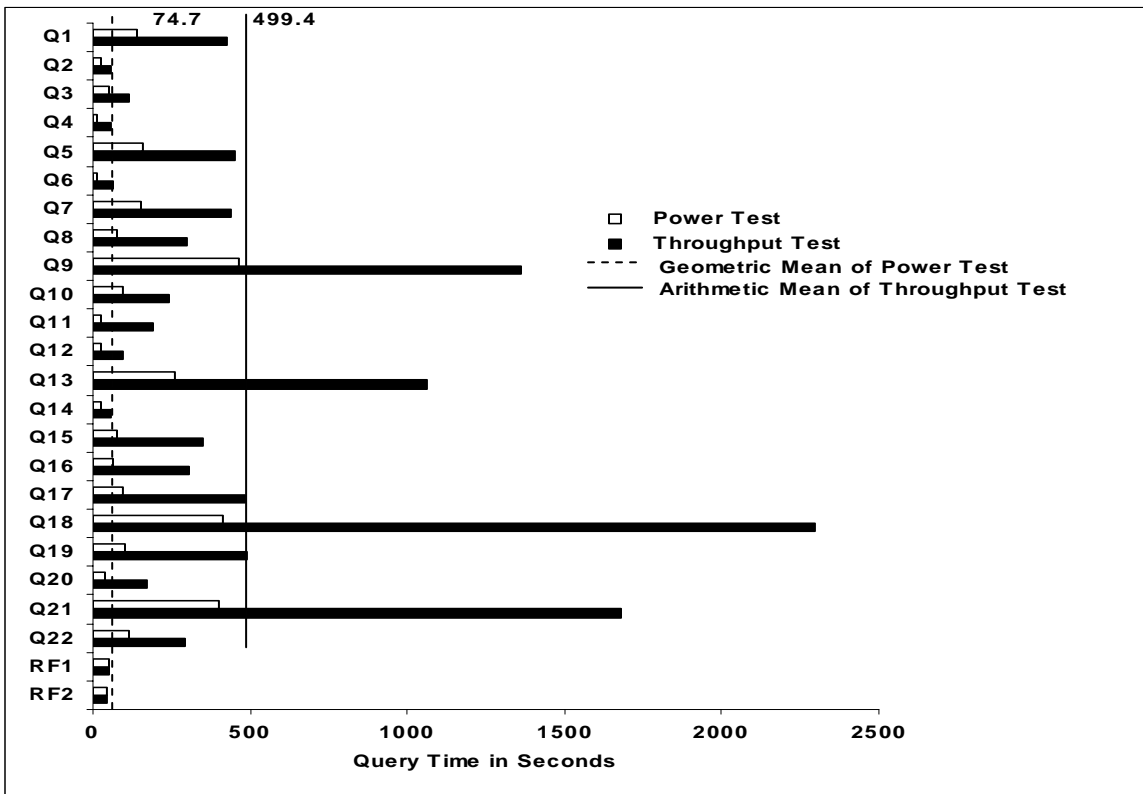
The following terms used in this publication are trademarks of other companies as follows: TPC Benchmark, TPC-H, QppH QthH and QphH are trademarks of Transaction Processing Performance Council; AMD and Opteron are trademarks or registered trademarks of AMD Corporation; Other company, product, or service names, which may be denoted by two asterisks (**), may be trademarks or service marks of others.

Notes

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

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

	IBM[®] @server[™] 325 with IBM DB2[®] UDB 8.1	TPC-H Revision 2.0	
		Report Date: July 29, 2003	
Total System Cost	Composite Query-per-hour Metric	Price/Performance	
863,410	13194.9 QphH@300GB	\$65 per QphH@300GB	
Database Size	Database Manager	Operating System	Availability Date
300GB	IBM DB2 UDB 8.1	SuSE Linux Enterprise Server 8	November 8, 2003



Database Load Time: 01:16:52	Load Included Backup: N	Total Data Storage / Database Size: 19.44
RAID (Base Table): Y	RAID (Base Tables and Auxiliary Data Structures): Y	RAID (ALL): Y

System Configuration: 8 nodes

- Processors:** 2 x 2GHz AMD 64-bit Opteron Model 246 processor on 2 processors, 2 cores, 2 threads
- Memory:** 6 x DIMMS 1GB ECC 2700 2.5 CL
- Disk Controllers:** 1 x QLA2342 HBA (2 ports x 2G bps FC)
1 x IBM FC2-133 HBA (1 ports x 2G bps FC)
- Disk Drives:** 3 (IBM TotalStorage FAS^tT EXP700 cases) × 14 (15K FC Hot-Swap Drive) × 18.2 GB
1 × 18.2GB 15K Ultra160 SCSI Hot-Swap Drive

Total Disk Storage: 5831.06 GB



IBM[®] @server[™] 325
with
IBM DB2[®] UDB 8.1

TPC-H Revision 2.0

Report Date:
July 29, 2003

Description	Part #	Source	Unit Price	Qty	Extended Price	3-Yr. Maint. Price
Server Hardware						
IBM e(logo)server 325 w/2GHz AMD 246	8835-51X		1 3,039	8	24,312	6968
2GHz Opteron Processor Upgrade	13N0700		1 1,399	8	11,192	0
DIMMS 1GB ECC 2700 2.5 CL	73P2267		1 659	48	31,632	0
Cisco Catalyst 3508G XL Enterprise edition	184837		3 3,600	1	3,600	1,223
Cisco 1000BASE-T Gigabit GBIC Module	372656		3 299	8	2,393	
18.2GB 15K Ultra160 SCSI Hot-Swap Drive	06P5767		1 329	8	2,632	0
IBM FC2-133 Host Bus Adapter	24P0960		1 1,485	8	11,880	0
Qlogic QLA 2342 Host Bus Adapter	408082		3 2,291	8	18,328	0
IBM Full size Keyboard PS/2	31P7415		1 45	1	45	
IBM Mouse	31P8700		1 20	1	20	
APC Smart-UPS Model 1400	32P1020		1 855	1	855	0
E54 15" (13.8" Viewable) Color Monitor	633147N		1 129	1	129	90
NetBAY42S Enterprise Rack	9306421		1 1,439	4	5,756	300
Subtotal					112,774	8581
Server Storage						
IBM TOTALSTORAGE FASTT EXP700	1740-1RU		1 3,000	24	72,000	17,280
18.2GB 15K FC Hot-Swap Drive	5211		1 681	336	228,816	
Short Wave SFP	2210		1 499	24	11,976	
1M LC-LC FIBRE OPTIC CABLE	5601		1 79	24	1,896	
Subtotal					300,816	17,280
Server Software						
DB2 UDB ESE V8.1 License/1-yr. Maint.	D518GLL		1 20,451	16	327,216	
DB2 UDB ESE V8.1 Support – 1Yr/Proc	E00BILL		1 974	32		31,168
DB2 UDB ESE V8.1 DPF License/1-yr. Maint.	D518JLL		1 6,143	16	98,288	
DB2 UDB ESE V8.1 DPF Support – 1Yr/Proc	E00BJLL		1 293	32		9,376
SuSE SLES8 8 proc + 1 yr maint.	2119-3INT-8		2 2,585	1	2,585	
SuSE SLES 8 Main. 1 yr/8proc	2119-MFJ-8		2 2,456	4		9,824
Subtotal					428,089	50,368
Large volume discount on IBM hardware of 14%; prices vary if purchased separately.14%					Discount	54,498
Total					787,181	76,229
Three-Year Cost of Ownership:						863,410
Pricing: 1 - IBM Business Partner; 2 - SuSE, 3 - CDW shop online Warranty and Maintenance: The standard warranty has been upgraded to 3 years of 24x7x4 coverage.					QphH:	13194.9
					\$/QphH:	\$65
Prices used in TPC benchmarks reflect the actual prices a customer would pay for a one-time purchase of the stated components. Individually negotiated discounts are not permitted. Special prices based on assumptions about past or future purchases are not permitted. All discounts reflect standard pricing policies for the listed components. For complete details, see the pricing sections of the TPC benchmark specifications. If you find that stated prices are not available according to these terms, please inform the TPC at pricing@tpc.org. Thank you.						



IBM® @server™ 325
with IBM DB2® UDB 8.1

TPC-H Revision 2.0
 Report Date: **July 29, 2003**

Measurement Results:

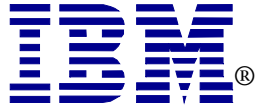
Database Scale Factor	300
Total Data Storage/Database Size	19.44
Start of Database Load	15:52:26
End of Database Load	17:09:18
Database Load Time	01:16:52
Query Streams for Throughput Test	6
TPC-H Power	14457.5
TPC-H Throughput	12042.6
TPC-H Composite Query-per-Hour (QphH@300GB)	13194.9
Total System Price over 3 Years	863,410
TPC-H Price/Performance Metric (\$/QphH@300GB)	65

Measurement Interval:

Measurement Interval in Throughput Test (Ts) = 11838 seconds

Duration of Stream Execution:

	Seed	Query Start Date/Time Query End Date/Time	RF1 Start Date/Time RF1 End Date/Time	RF2 Start Date/Time RF2 End Date/Time	Duration
Stream 00	727170918	07/27/03 22:03:30	07/27/03 22:02:40	07/27/03 22:50:30	00:48:36
		07/27/03 22:50:30	07/27/03 22:03:30	07/27/03 22:51:15	
Stream 01	727170919	07/27/03 22:51:20	07/28/03 01:59:11	07/28/03 02:00:06	03:07:51
		07/28/03 01:59:11	07/28/03 02:00:06	07/28/03 02:00:53	
Stream 02	727170920	07/27/03 22:51:21	07/28/03 02:00:53	07/28/03 02:01:45	03:06:53
		07/28/03 01:58:14	07/28/03 02:01:45	07/28/03 02:02:32	
Stream 03	727170921	07/27/03 22:51:20	07/28/03 02:02:32	07/28/03 02:03:21	03:02:32
		07/28/03 01:53:52	07/28/03 02:03:21	07/28/03 02:04:05	
Stream 04	727170922	07/27/03 22:51:21	07/28/03 02:04:05	07/28/03 02:04:54	03:01:52
		07/28/03 01:53:13	07/28/03 02:04:54	07/28/03 02:05:37	
Stream 05	727170923	07/27/03 22:51:21	07/28/03 02:05:37	07/28/03 02:06:25	03:00:18
		07/28/03 01:51:39	07/28/03 02:04:54	07/28/03 02:07:08	
Stream 06	727170924	07/27/03 22:51:22	07/28/03 02:07:08	07/28/03 02:07:55	02:59:16
		07/28/03 01:50:38	07/28/03 02:07:55	07/28/03 02:08:39	



IBM® *e*server™ 325
with IBM DB2® UDB 8.1

TPC-H Revision 2.0

Report Date: July 29, 2003

TPC-H Timing Intervals (in seconds):

Query	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
Stream 00	139.4	25.1	52.6	15.1	160.5	15.5	151.0	73.7	465.8	94.1	24.9	27.1
Stream 01	179.7	67.8	95.2	16.4	423.1	69.8	530.4	291.2	1707.1	182.6	193.4	81.8
Stream 02	562.9	73.0	161.1	77.1	421.9	59.2	399.4	266.3	1683.2	244.1	187.9	87.9
Stream 03	510.9	49.5	82.8	83.3	445.4	70.1	350.5	388.3	1161.0	181.3	237.6	74.6
Stream 04	501.6	50.9	124.9	43.4	502.7	58.1	496.2	276.6	1213.1	254.3	249.4	106.2
Stream 05	383.3	53.3	109.5	52.0	463.7	60.2	462.1	287.4	1232.8	246.4	161.7	108.5
Stream 06	425.2	55.9	101.5	52.8	469.8	59.1	389.7	273.6	1165.1	337.9	130.4	119.8
Minimum	179.7	49.5	82.8	16.4	421.9	58.1	350.5	266.3	1161.0	181.3	130.4	74.6
Average	427.3	58.4	112.5	54.2	454.4	62.8	438.1	297.2	1360.4	241.1	193.4	96.5
Maximum	562.9	73.0	161.1	83.3	502.7	70.1	530.4	388.3	1707.1	337.9	249.4	119.8

Stream ID	Q13	Q14	Q15a	Q16	Q17	Q18	Q19	Q20	Q21	Q22	RF1	RF2
Stream 00	259.6	23.8	74.2	63.2	93.5	415.6	100.6	35.4	397.9	111.4	50.1	45.0
Stream 01	1195.4	52.5	426.3	251.0	413.0	2346.5	537.5	174.6	1880.2	155.6	55.0	47.0
Stream 02	1039.1	55.7	319.8	384.2	489.6	2425.4	619.7	257.2	985.5	413.0	51.5	47.4
Stream 03	1010.0	58.1	419.8	300.9	570.2	2260.3	532.2	186.0	1590.0	389.2	48.5	44.5
Stream 04	1104.5	64.4	240.6	293.4	445.4	2230.3	374.0	94.5	1908.1	279.1	48.3	43.8
Stream 05	950.8	68.6	377.5	249.6	553.5	2275.6	488.6	157.8	1841.5	233.8	47.3	43.7
Stream 06	1075.9	48.0	307.9	352.1	423.1	2250.8	391.1	160.2	1869.4	296.9	46.4	43.6
Minimum	950.8	48.0	240.6	249.6	413.0	2230.3	374.0	94.5	985.5	155.6	46.4	43.6
Average	1062.6	57.9	348.7	305.2	482.5	2298.2	490.5	171.7	1679.1	294.6	49.5	45.0
Maximum	1195.4	68.6	426.3	384.2	570.2	2425.4	619.7	257.2	1908.1	413.0	55.0	47.4

Benchmark Sponsor: Haider Rizvi
 Mgr., DB2 Data Warehouse
 Performance
 IBM Canada Ltd;
 8200 Warden Avenue
 Markham, Ontario L6G 1C7

July 29, 2003

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

Platform: **IBM @server eServer 335 8-node cluster**
 Database Manager: **IBM DB2 UDB 8.1 ESE with DPF**
 Operating System: **SuSE Linux 8.1**

The results were:

CPU (Speed)	Memory	Disks	QphH@300GB
IBM @server eServer 335 (each node with)			
2 x AMD 64-bit Opteron (2.0 GHz)	1 MB Cache 6 GB Main	42 x 18.2 GB FASTt 1 x 18.2 GB SCSI	13,194.9

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

- The database records were defined with the proper layout and size
- The database population was generated using DBGEN
- The database was properly scaled to 300GB and populated accordingly
- The compliance of the database auxiliary data structures was verified
- The database load time was correctly measured and reported

- The required ACID properties were verified and met
- The query input variables were generated by QGEN
- The query text was produced using minor modifications and an approved query variant
- The execution of the queries against the SF1 database produced compliant answers
- A compliant implementation specific layer was used to drive the tests
- The throughput tests involved 6 query streams
- The ratio between the longest and the shortest query was such that no query timing was adjusted
- The execution times for queries and refresh functions were correctly measured and reported
- The repeatability of the measured results was verified
- The required amount of database log was configured
- The system pricing was verified for major components and maintenance
- The major pages from the FDR were verified for accuracy

Additional Audit Notes:

None.

Respectfully Yours,

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

François Raab
President

Table of Contents

Preface	12
1 General Items	14
1.1 Benchmark Sponsor	14
1.2 Parameter Settings	14
1.3 Configuration Diagrams	14
1.3.1 Measured and Priced Configurations	15
2 Clause 1: Logical Database Design Related Items	16
2.1 Database Table Definitions	16
2.2 Database Organization	16
2.3 Horizontal Partitioning	16
2.4 Replication	16
3 Clause 2: Queries and Update Functions Related Items	17
3.1 Query Language	17
3.2 Random Number Generation	17
3.3 Substitution Parameters Generation	17
3.4 Query Text and Output Data from Database	17
3.5 Query Substitution Parameters and Seeds Used	17
3.6 Query Isolation Level	17
3.7 Refresh Function Implementation	18
4 Clause 3: Database System Properties Related Items	19
4.1 Atomicity Requirements	19
4.1.1 Atomicity of Completed Transactions	19
4.1.2 Atomicity of Aborted Transactions	19
4.2 Consistency Requirements	19
4.2.1 Consistency Condition	19
4.2.2 Consistency Tests	20
4.3 Isolation Requirements	20
4.3.1 Isolation Test 1	20
4.3.2 Isolation Test 2	20
4.3.3 Isolation Test 3	20
4.3.4 Isolation Test 4	21
4.3.5 Isolation Test 5	21

4.3.6 Isolation Test 6	22
4.4 Durability Requirements	22
4.4.1 Failure of Durable Medium Containing Recovery Log Data, and Loss of System Power/Memory	22
4.4.2 Loss of Switch Power	23
5 Clause 4: Scaling and Database Population Related Items	24
5.1 Cardinality of Tables	24
5.2 Distribution of Tables and Logs	24
5.3 Database Partition / Replication Mapping	25
5.4 RAID Implementation	25
5.5 DBGEN Modifications	25
5.6 Database Load Time	25
5.7 Data Storage Ratio	25
5.8 Database Load Mechanism Details and Illustration	25
5.9 Qualification Database Configuration	26
6 Clause 5: Performance Metrics and Execution Rules Related Items	27
6.1 System Activity between Load and Performance Tests	27
6.2 Steps in the Power Test	27
6.3 Timing Intervals for Each Query and Refresh Function	27
6.4 Number of Streams for the Throughput Test	27
6.5 Start and End Date/Times for Each Query Stream	27
6.6 Total Elapsed Time for the Measurement Interval	27
6.7 Refresh Function Start Date/Time and Finish Date/Time	27
6.8 Timing Intervals for Each Query and Each Refresh Function for Each Stream	28
6.9 Performance Metrics	28
6.10 Performance Metric and Numerical Quantities from Both Runs	28
6.11 System Activity between Tests	28
7 Clause 6: SUT and Driver Implementation Related Items	29
7.1 Driver	29
7.2 Implementation-Specific Layer	29
7.3 Profile-Directed Optimization	29
8 Clause 7: Pricing Related Items	30
8.1 Hardware and Software Components	30
8.2 Three-Year Cost of System Configuration	30
8.3 Availability Dates	30

8.4 Country-Specific Pricing	30
9 Clause 8: Audit Related Items	31
9.1 Auditor's Report	31
Appendix A: Tunable Parameters and System Configuration	32
DB2 UDB 8.1 Database and Database Manager Configuration	32
<i>Database Configuration for Node 0</i>	32
<i>Database Configuration for Node 1</i>	33
<i>Database Configuration for Node 2</i>	35
<i>Database Configuration for Node 3</i>	37
<i>Database Configuration for Node 4</i>	38
<i>Database Configuration for Node 5</i>	40
<i>Database Configuration for Node 6</i>	41
<i>Database Configuration for Node 7</i>	43
<i>Database Configuration for Node 8</i>	45
<i>Database Configuration for Node 9</i>	46
<i>Database Configuration for Node 10</i>	48
<i>Database Configuration for Node 11</i>	50
<i>Database Configuration for Node 12</i>	51
<i>Database Configuration for Node 13</i>	53
<i>Database Configuration for Node 14</i>	54
<i>Database Configuration for Node 15</i>	56
DB2 Database Manager Configuration	58
DB2 Registry Variables	59
Linux Parameters	59
Appendix B: Database Build Scripts	60
buildtpcd	60
create_bufferpools	74
create_indexes	74
create_nodegroups	75
create_tables	75
create_tablespaces	77
createuftbls	82
db2nodes.cfg	83
Load_db2set.ksh	83

run_db2set.ksh	83
runstats.ddl	83
load_tables.ksh	84
load_tb_customer.ddl	85
load_tb_lineitem.ddl	85
load_tb_orders.ddl	85
load_tb_part.ddl	85
load_tb_partsupp.ddl	85
load_tb_supplier.ddl	86
Load_dbcfg.ddl	86
load_dbmcfg.ddl	86
run_dbcfg.ddl	86
run_dbmcfg.ddl	86
Change_logs.ksh	86
tpcd.setup	87
Appendix C: Qualification Query Output	92
Qualification Queries	92
<i>Query 1</i>	92
<i>Query 2</i>	92
<i>Query 3</i>	97
<i>Query 4</i>	98
<i>Query 5</i>	98
<i>Query 6</i>	99
<i>Query 7</i>	99
<i>Query 8</i>	100
<i>Query 9</i>	100
<i>Query 10</i>	103
<i>Query 11</i>	105
<i>Query 12</i>	105
<i>Query 13</i>	106
<i>Query 14</i>	107
<i>Query 15a</i>	107
<i>Query 16</i>	108
<i>Query 17</i>	108

<i>Query 18</i>	109
<i>Query 19</i>	110
<i>Query 20</i>	111
<i>Query 21</i>	115
<i>Query 22</i>	116
First 10 Rows of the Database	117
Query Substitution Parameters	121
Appendix D: Driver Source Code	126
ploaduf1	126
ploaduf2	126
load_line_uf	126
load_orders_uf	126
preload_uf.ksh	126
runpower	126
runthroughput	131
tpcdbatch.h	135
tpcdbatch.sqc	137
tpcdUF.sqc	182
Appendix E: ACID Transaction Source Code	191
acid.h	191
acid.sqc	191
makefile	206
Appendix F: Price Quotations	207

Preface

TPC Benchmark H Standard Specification was developed by the Transaction Processing Performance Council (TPC). It was released on February 26, 1999, and most recently revised (Revision 2.0) October 29, 2002. This is the full disclosure report for benchmark testing of IBM **@server** 325 according to the TPC Benchmark H Standard Specification.

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

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

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

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

The TPC-H operations are modeled as follows:

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

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

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

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

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

1 General Items

1.1 Benchmark Sponsor

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

IBM Corporation sponsored this TPC-H benchmark.

1.2 Parameter Settings

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

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

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

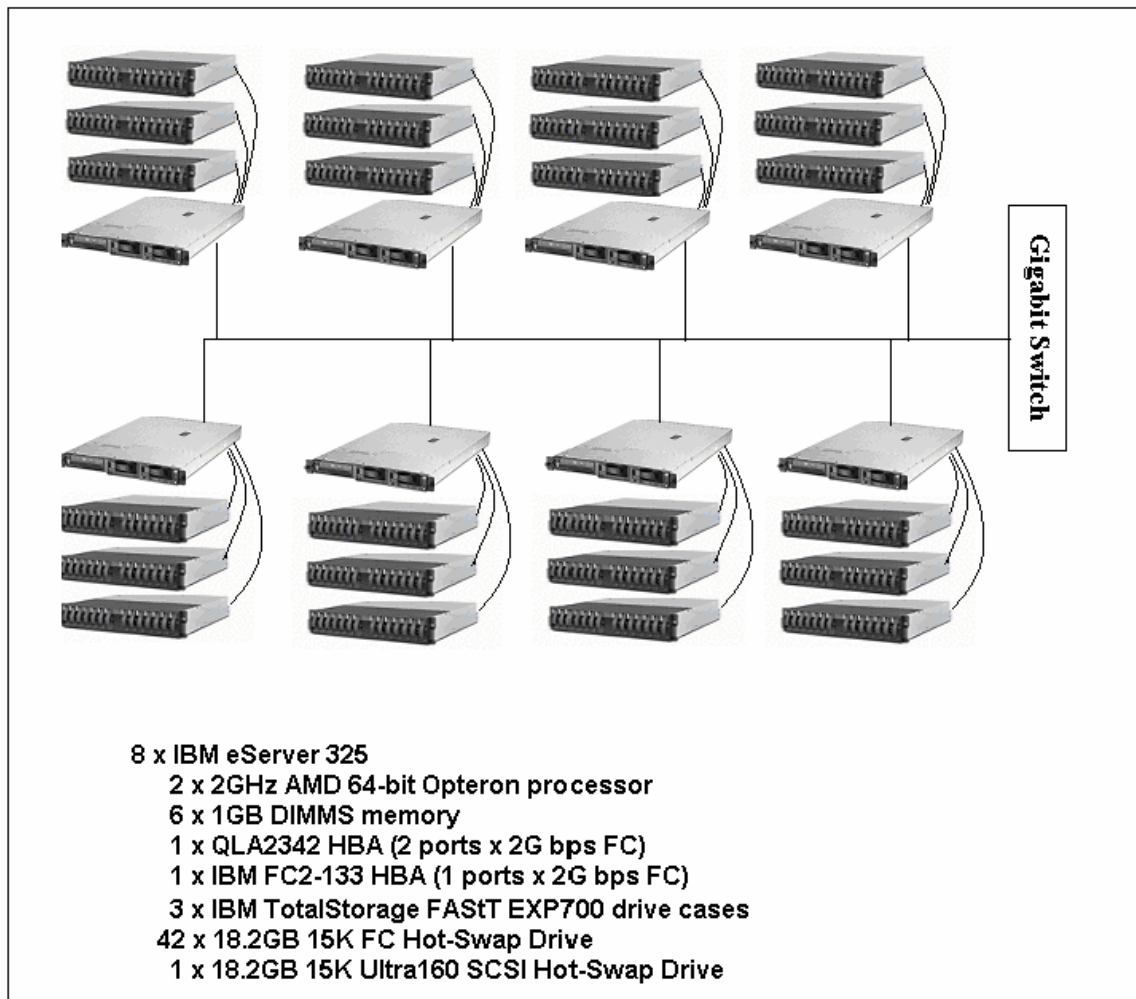
1.3 Configuration Diagrams

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

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

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

1.3.1 Priced and Measured Configurations



The priced configuration was an 8 x IBM **e**server 325 each with

- Two 2GHz AMD 64-bit Opteron Model 246 processors
- 6 GB of memory
- One QLA2342 HBA (2 ports x 2G bps FC)
- One IBM FC2-133 HBA (1 port x 2G bps FC)
- One 18.2GB 15K Ultra160 SCSI disk drive
- Forty-two 18.2GB 15K FC Hot-swap Drives
- Three IBM TotalStorage FAStT EXP700 drive cases

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

2 Clause 1: Logical Database Design Related Items

2.1 Database Table Definitions

Listings must be provided for all table definition statements and all other statements used to set up the test and qualification databases. (8.1.2.1)

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

2.2 Database Organization

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

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

2.3 Horizontal Partitioning

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

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

2.4 Replication

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

No replication was used.

3 Clause 2: Queries and Update Functions Related Items

3.1 Query Language

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

SQL was the query language used.

3.2 Random Number Generation

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

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

3.3 Substitution Parameters Generation

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

The supplied QGEN version 1.3.0 was used to generate the substitution parameters.

3.4 Query Text and Output Data from Database

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

Appendix C.1, “Qualification Queries,” contains the output for each of the queries. The functional query definitions and variants used in this disclosure use the following minor query modifications:

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

3.5 Query Substitution Parameters and Seeds Used

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

Appendix C contains the seed and query substitution parameters used.

3.6 Query Isolation Level

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

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

3.7 Refresh Function Implementation

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

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

4 Clause 3: Database System Properties Related Items

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

All ACID tests were conducted according to specifications. The Atomicity, Isolation, Consistency and Durability tests were performed on the IBM **@server** 325. Appendix E contains the ACID transaction source code.

4.1 Atomicity Requirements

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

4.1.1 Atomicity of Completed Transactions

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

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

1. The total price from the ORDER table and the extended price from the LINEITEM table were retrieved for a randomly selected order key. The number of records in the HISTORY table was also retrieved.
2. The ACID Transaction T1 was executed for the order key used in step 1.
3. The total price and extended price were retrieved for the same order key used in step 1 and step 2.

It was verified that:

$$T1.EXTENDEDPRICE=OLD.EXTENDEDPRICE+((T1.DELTA)*$$
$$(OLD.EXTENDEDPRICE/OLD.QUANTITY)),T1.TOTALPRICE=OLD.TOTALPRICE+$$
$$((T1.EXTENDEDPRICE-OLD.EXTENDEDPRICE)*(1-DISCOUNT)*(1+TAX)),$$

and that the number of records in the History table had increased by 1.

4.1.2 Atomicity of Aborted Transactions

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

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

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

4.2 Consistency Requirements

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

4.2.1 Consistency Condition

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

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

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

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

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

4.2.2 Consistency Tests

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

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

After executing 7 streams of 100 ACID transactions each, the queries defined in 4.2.1, “Consistency Condition,” were run again. The queries showed that the database was still in a consistent state.

4.3 Isolation Requirements

4.3.1 Isolation Test 1

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

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

1. First session: Start an ACID transaction with a randomly selected O_KEY, L_KEY and DELTA. The transaction is delayed for 60 seconds just prior to the Commit.
2. Second session: Start an ACID query for the same O_KEY as in the ACID transaction.
3. Second session: The ACID query attempts to read the file but is locked out by the ACID transaction waiting to complete.
4. First session: The ACID transaction is released and the Commit is executed releasing the record. With the LINEITEM record now released, the ACID query can now complete.
5. Second session: Verify that the ACID query delays for approximately 60 seconds and that the results displayed for the ACID query match the input for the ACID transaction.

4.3.2 Isolation Test 2

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

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

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

4.3.3 Isolation Test 3

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

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

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

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

4.3.4 Isolation Test 4

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

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

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

4.3.5 Isolation Test 5

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

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

4.3.6 Isolation Test 6

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

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

4.4 Durability Requirements

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

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

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

The database log was stored on RAID-1 protected storage. The tables for the database were stored on RAID-1 protected storage. A backup of the database was taken to a separate array for drives than those used for the database.

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

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

4.4.2. Loss of Switch Power

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

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

5 Clause 4: Scaling and Database Population Related Items

5.1 Cardinality of Tables

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

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

5.2 Distribution of Tables and Logs

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

DB2 was configured on an 8 node IBM eServer 325 test system. Each node had:

- 1 IBM FC2-133 Host Bus Adapter and 1 Qlogic QLA 2342 Host Bus Adapter.
- 3 (IBM TotalStorage FAStT EXP700 enclosures) × 14 (15K FC Hot-Swap Drive) × 18.2 GB and 1 18.2GB 15K Ultra160 SCSI Hot-Swap Drive internal disk drive.

Each e325 had 22 RAID 1 volumes. 20 of these volumes were created using linux software raid provided in SuSE Linux Enterprise Server 8 on 40 partitions across the 3 EXP700 enclosures so that there were 7 volumes on the first 2 enclosures and 6 volumes on the third enclosure. 2 addition RAID 1 volumes were created on the third enclosure by mirroring 4 partitions that were created on the remaining 2 drives.

The tablespace for NATION and REGION were stored on a single RAID volume only on node 1.

For each node, 20 RAID volumes were used for the table and index data tablespace(DATA_INDEX), the temp space tablespaces (TEMPSPACE)were placed on the same 40 disks on separate partitions shared by the table and index tablespace , database logs were placed on the remaining 2 RAID devices.

The staging data for RF functions as well as data flatfiles were on 12 partitions of the last exp 700 enclosure protected by RAID 5.

The Operating System resided on an internal disk on each node. The benchmark executing programs resided on a shared home directory NFS mounted from node 1. The database software resided on an internal disk on each node.

5.3 Database Partition / Replication Mapping

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

The database was not replicated. The database was logically partitioned into 16 logical nodes, 2 nodes on each physical server.

5.4 RAID Implementation

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

RAID level 1 was used across all database tables, indexes, and recovery logs through linux software raid provided with SuSE Linux Enterprise Server 8.

5.5 DBGEN Modifications

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

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

5.6 Database Load Time

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

See the Executive Summary at the beginning of this report.

5.7 Data Storage Ratio

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

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

Disk Type	Number of Disks	Formatted Space per Disk	Total Disk Space	Scale Factor	Storage Ratio
18.2GB 15K Ultra160 SCSI Drive	336	16.95GB	5695.45GB		
18.2GB 15K Ultra160 SCSI Hot-Swap Drive	8	16.95GB	135.61GB		
Total			5831.06GB	300	19.44

The data storage ratio is 19.44, derived by dividing 5831.06GB by the database size of 300GB.

5.8 Database Load Mechanism Details and Illustration

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

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

The NATION and REGION tables were created on node 0 and then loaded from dbgen output. The other tables were loaded on 16 logical nodes.

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

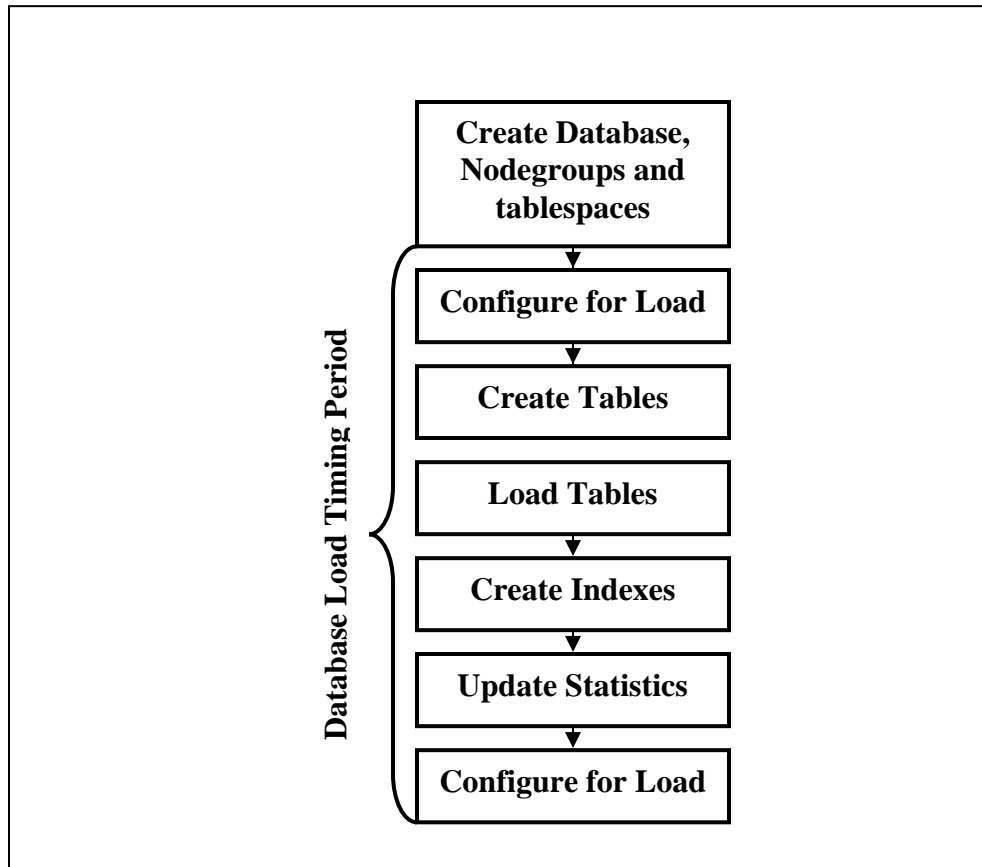


Figure 4-1. Database Load Procedure

5.9 Qualification Database Configuration

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

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

6 Clause 5: Performance Metrics and Execution Rules Related Items

6.1 System Activity between Load and Performance Tests

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

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

6.2 Steps in the Power Test

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

The following steps were used to implement the power test:

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

6.3 Timing Intervals for Each Query and Refresh Function

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

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

6.4 Number of Streams for the Throughput Test

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

Five streams were used for the throughput test.

6.5 Start and End Date/Times for Each Query Stream

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

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

6.6 Total Elapsed Time for the Measurement Interval

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

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

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

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

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

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

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

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

6.9 Performance Metrics

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

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

6.10 Performance Metric and Numerical Quantities from Both Runs

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

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

	QppH @ 300GB	QthH @ 300GB	QphH @ 300GB
Run1	14833.4	12000.0	13341.7
Run2	14457.5	12042.6	13194.9

6.11 System Activity between Tests

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

DB2 was restarted between runs.

7 Clause 6: SUT and Driver Implementation Related Items

7.1 Driver

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

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

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

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

7.2 Implementation-Specific Layer

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

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

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

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

7.3 Profile-Directed Optimization

Profile-directed optimization was not used.

8 Clause 7: Pricing Related Items

8.1 Hardware and Software Components

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

A detailed list of all hardware and software, including the 3-year price, is provided in the Executive Summary at the front of this report. The price quotations are included in Appendix F.

8.2 Three-Year Cost of System Configuration

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

A detailed list of all hardware and software, including the 3-year price, is provided in the Executive Summary at the front of this report. The price quotations are included in Appendix F.

8.3 Availability Dates

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

The system as priced will be generally available November 8, 2003.

8.4 Country-Specific Pricing

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

The configuration is priced for the United States of America.

9 Clause 8: Audit Related Items

9.1 Auditor's Report

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

This implementation of the TPC Benchmark H was audited by Francois Raab of InfoSizing, Inc. Further information can be downloaded from www.tpc.org.

Appendix A: Tunable Parameters and System Configuration

DB2 UDB 8.1 Database and Database Manager Configuration

Database Configuration for Node 0

Database Configuration for Database TPCD		Data Links Write Token Init Expiry Intvl(DL_WT_IEXPINT) = 60
Database configuration release level	=	Data Links Number of Copies (DL_NUM_COPIES) = 1
0x0a00		Data Links Time after Drop (days) (DL_TIME_DROP) = 1
Database release level	=	Data Links Token in Uppercase (DL_UPPER) = NO
0x0a00		Data Links Token Algorithm (DL_TOKEN) = MAC0
Database territory	= US	Database heap (4KB) (DBHEAP)
Database code page	= 819	= 10000
Database code set	=	Size of database shared memory (4KB) (DATABASE_MEMORY) = AUTOMATIC
ISO8859-1		Catalog cache size (4KB) (CATALOGCACHE_SZ) = (MAXAPPLS*4)
Database country/region code	= 1	Log buffer size (4KB) (LOGBUFSZ)
		= 2048
Dynamic SQL Query management (DYN_QUERY_MGMT) = DISABLE		Utilities heap size (4KB) (UTIL_HEAP_SZ) = 5000
Discovery support for this database (DISCOVER_DB) = ENABLE		Buffer pool size (pages) (BUFFPAGE)
Default query optimization class (DFT_QUERYOPT) = 7		= 1000
Degree of parallelism (DFT_DEGREE) = 1		Extended storage segments size (4KB) (ESTORE_SEG_SZ) = 16000
Continue upon arithmetic exceptions (DFT_SQLMATHWARN) = NO		Number of extended storage segments (NUM_ESTORE_SEGS) = 0
Default refresh age (DFT_REFRESH_AGE) = 0		Max storage for lock list (4KB) (LOCKLIST) = 40000
Number of frequent values retained (NUM_FREQVALUES) = 0		Max size of appl. group mem set (4KB) (APPGROUP_MEM_SZ) = 2000
Number of quantiles retained (NUM_QUANTILES) = 300		Percent of mem for appl. group heap (GROUPHEAP_RATIO) = 70
Backup pending	= NO	Max appl. control heap size (4KB) (APP_CTL_HEAP_SZ) = 512
Database is consistent	= YES	Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = 250
Rollforward pending	= NO	Sort list heap (4KB) (SORTHEAP) = 20000
Restore pending	= NO	SQL statement heap (4KB) (STMTHEAP) = 10000
Multi-page file allocation enabled	=	Default application heap (4KB) (APPLHEAPSZ) = 1024
NO		Package cache size (4KB) (PCKCACHESZ) = (MAXAPPLS*8)
Log retain for recovery status	= NO	Statistics heap size (4KB) (STAT_HEAP_SZ) = 4384
User exit for logging status	= NO	Interval for checking deadlock (ms) (DLCHKTIME) = 10000
Data Links Token Expiry Interval (sec) (DL_EXPINT) = 60		Percent. of lock lists per application (MAXLOCKS) = 20
		Lock timeout (sec) (LOCKTIMEOUT) = -1

Changed pages threshold (CHNGPGS_THRESH) = 60		User exit for logging enabled (USEREXIT) = OFF
Number of asynchronous page cleaners (NUM_IOCLEANERS) = 4		Auto restart enabled (AUTORESTART) = ON
Number of I/O servers (NUM_IOSERVERS) = 4		Index re-creation time (INDEXREC) = SYSTEM (RESTART)
Index sort flag (INDEXSORT) = YES		Default number of loadrec sessions (DFT_LOADREC_SES) = 1
Sequential detect flag (SEQDETECT) = YES		Number of database backups to retain (NUM_DB_BACKUPS) = 12
Default prefetch size (pages) (DFT_PREFETCH_SZ) = 32		Recovery history retention (days) (REC_HIS_RETENTN) = 366
Track modified pages (TRACKMOD) = OFF		TSM management class (TSM_MGMTCLASS) =
Default number of containers = 1		TSM node name (TSM_NODENAME) =
Default tablespace extentsize (pages) (DFT_EXTENT_SZ) = 32		TSM owner (TSM_OWNER) =
Max number of active applications (MAXAPPLS) = AUTOMATIC		TSM password (TSM_PASSWORD) =
Average number of active applications (AVG_APPLS) = 1		
Max DB files open per application (MAXFILOP) = 1024		
Log file size (4KB) (LOGFILSIZ) = 50000		
Number of primary log files (LOGPRIMARY) = 4		
Number of secondary log files (LOGSECOND) = 1		
Changed path to log files (NEWLOGPATH) =		
Path to log files = /dev/raw/raw61		
Overflow log path (OVERFLOWLOGPATH) =		
Mirror log path (MIRRORLOGPATH) =		
First active log file =		
Block log on disk full (BLK_LOG_DSK_FUL) = NO		
Percent of max active log space by transaction (MAX_LOG) = 0		
Num. of active log files for 1 active UOW (NUM_LOG_SPAN) = 0		
Group commit count (MINCOMMIT) = 2		
Percent log file reclaimed before soft chckpt (SOFTMAX) = 360		
Log retain for recovery enabled (LOGRETAIN) = OFF		
		Database Configuration for Node 1
		Database Configuration for Database TPCD
		Database configuration release level = 0x0a00
		Database release level = 0x0a00
		Database territory = US
		Database code page = 819
		Database code set = ISO8859-1
		Database country/region code = 1
		Dynamic SQL Query management (DYN_QUERY_MGMT) = DISABLE
		Discovery support for this database (DISCOVER_DB) = ENABLE
		Default query optimization class (DFT_QUERYOPT) = 7
		Degree of parallelism (DFT_DEGREE) = 1
		Continue upon arithmetic exceptions (DFT_SQLMATHWARN) = NO
		Default refresh age (DFT_REFRESH_AGE) = 0
		Number of frequent values retained (NUM_FREQVALUES) = 0
		Number of quantiles retained (NUM_QUANTILES) = 300

Backup pending	= NO	SQL statement heap (4KB) (STMTHEAP) = 10000
Database is consistent	= YES	Default application heap (4KB) (APPLHEAPSZ) = 1024
Rollforward pending	= NO	Package cache size (4KB)
Restore pending	= NO	(PCKCACHESZ) = (MAXAPPLS*8)
Multi-page file allocation enabled	=	Statistics heap size (4KB) (STAT_HEAP_SZ) = 4384
NO		Interval for checking deadlock (ms) (DLCHKTIME) = 10000
Log retain for recovery status	= NO	Percent. of lock lists per application
User exit for logging status	= NO	(MAXLOCKS) = 20
		Lock timeout (sec) (LOCKTIMEOUT) = -1
Data Links Token Expiry Interval (sec) (DL_EXPINT) = 60		Changed pages threshold (CHNGPGS_THRESH) = 60
Data Links Write Token Init Expiry Intvl(DL_WT_IEXPINT) = 60		Number of asynchronous page cleaners (NUM_IOCLEANERS) = 4
Data Links Number of Copies (DL_NUM_COPIES) = 1		Number of I/O servers (NUM_IOSERVERS) = 4
Data Links Time after Drop (days) (DL_TIME_DROP) = 1		Index sort flag (INDEXSORT) =
Data Links Token in Uppercase (DL_UPPER) = NO		YES
Data Links Token Algorithm (DL_TOKEN) = MAC0		Sequential detect flag (SEQDETECT) = YES
Database heap (4KB) (DBHEAP) = 10000		Default prefetch size (pages) (DFT_PREFETCH_SZ) = 32
Size of database shared memory (4KB) (DATABASE_MEMORY) = AUTOMATIC		Track modified pages (TRACKMOD) = OFF
Catalog cache size (4KB) (CATALOGCACHE_SZ) = (MAXAPPLS*4)		Default number of containers = 1
Log buffer size (4KB) (LOGBUFSZ) = 2048		Default tablespace extentsize (pages) (DFT_EXTENT_SZ) = 32
Utilities heap size (4KB) (UTIL_HEAP_SZ) = 5000		Max number of active applications (MAXAPPLS) = AUTOMATIC
Buffer pool size (pages) (BUFFPAGE) = 1000		Average number of active applications (AVG_APPLS) = 1
Extended storage segments size (4KB) (ESTORE_SEG_SZ) = 16000		Max DB files open per application (MAXFILOP) = 1024
Number of extended storage segments (NUM_ESTORE_SEGS) = 0		Log file size (4KB) (LOGFILSIZ) =
Max storage for lock list (4KB) (LOCKLIST) = 40000		50000
Max size of appl. group mem set (4KB) (APPGROUP_MEM_SZ) = 2000		Number of primary log files (LOGPRIMARY) = 4
Percent of mem for appl. group heap (GROUPHEAP_RATIO) = 70		Number of secondary log files (LOGSECOND) = 1
Max appl. control heap size (4KB) (APP_CTL_HEAP_SZ) = 512		Changed path to log files (NEWLOGPATH) =
Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = 250		Path to log files =
Sort list heap (4KB) (SORTHEAP) =		/dev/raw/raw62
20000		Overflow log path (OVERFLOWLOGPATH) =
		Mirror log path (MIRRORLOGPATH) =

First active log file	=	Discovery support for this database	
Block log on disk full		(DISCOVER_DB) = ENABLE	
(BLK_LOG_DSK_FUL) = NO		Default query optimization class	
Percent of max active log space by		(DFT_QUERYOPT) = 7	
transaction(MAX_LOG) = 0		Degree of parallelism	
Num. of active log files for 1 active		(DFT_DEGREE) = 1	
UOW(NUM_LOG_SPAN) = 0		Continue upon arithmetic exceptions	
		(DFT_SQLMATHWARN) = NO	
Group commit count		Default refresh age	
(MINCOMMIT) = 2		(DFT_REFRESH_AGE) = 0	
Percent log file reclaimed before soft ckcpt		Number of frequent values retained	
(SOFTMAX) = 360		(NUM_FREQVALUES) = 0	
Log retain for recovery enabled		Number of quantiles retained	
(LOGRETAIN) = OFF		(NUM_QUANTILES) = 300	
User exit for logging enabled			
(USEREXIT) = OFF			
		Backup pending	= NO
Auto restart enabled		Database is consistent	= YES
(AUORESTART) = ON		Rollforward pending	= NO
Index re-creation time	(INDEXREC)	Restore pending	= NO
= SYSTEM (RESTART)			
Default number of loadrec sessions		Multi-page file allocation enabled	=
(DFT_LOADREC_SES) = 1		NO	
Number of database backups to retain			
(NUM_DB_BACKUPS) = 12		Log retain for recovery status	= NO
Recovery history retention (days)		User exit for logging status	= NO
(REC_HIS_RETENTN) = 366			
		Data Links Token Expiry Interval (sec)	
TSM management class		(DL_EXPINT) = 60	
(TSM_MGMTCLASS) =		Data Links Write Token Init Expiry	
TSM node name		Intvl(DL_WT_IEXPINT) = 60	
(TSM_NODENAME) =		Data Links Number of Copies	
TSM owner	(TSM_OWNER)	(DL_NUM_COPIES) = 1	
=		Data Links Time after Drop (days)	
TSM password		(DL_TIME_DROP) = 1	
(TSM_PASSWORD) =		Data Links Token in Uppercase	
		(DL_UPPER) = NO	
		Data Links Token Algorithm	
		(DL_TOKEN) = MAC0	
		Database heap (4KB)	(DBHEAP)
		= 10000	
		Size of database shared memory (4KB)	
		(DATABASE_MEMORY) = AUTOMATIC	
		Catalog cache size (4KB)	
		(CATALOGCACHE_SZ) = (MAXAPPLS*4)	
		Log buffer size (4KB)	(LOGBUFSZ)
		= 2048	
		Utilities heap size (4KB)	
		(UTIL_HEAP_SZ) = 5000	
		Buffer pool size (pages)	(BUFFPAGE)
		= 1000	
		Extended storage segments size (4KB)	
		(ESTORE_SEG_SZ) = 16000	

Database Configuration for Node 2

Database Configuration for Database TPCD

Database configuration release level	=
0x0a00	
Database release level	=
0x0a00	
Database territory	= US
Database code page	= 819
Database code set	=
ISO8859-1	
Database country/region code	= 1
Dynamic SQL Query management	
(DYN_QUERY_MGMT) = DISABLE	

Number of extended storage segments (NUM_ESTORE_SEGS) = 0	Max DB files open per application (MAXFILOP) = 1024
Max storage for lock list (4KB) (LOCKLIST) = 40000	Log file size (4KB) (LOGFILSIZ) = 50000
Max size of appl. group mem set (4KB) (APPGROUP_MEM_SZ) = 2000	Number of primary log files (LOGPRIMARY) = 4
Percent of mem for appl. group heap (GROUPHEAP_RATIO) = 70	Number of secondary log files (LOGSECOND) = 1
Max appl. control heap size (4KB) (APP_CTL_HEAP_SZ) = 512	Changed path to log files (NEWLOGPATH) =
Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = 250	Path to log files =
Sort list heap (4KB) (SORTHEAP) = 20000	/dev/raw/raw61
SQL statement heap (4KB) (STMTHEAP) = 10000	Overflow log path (OVERFLOWLOGPATH) =
Default application heap (4KB) (APPLHEAPSZ) = 1024	Mirror log path (MIRRORLOGPATH) =
Package cache size (4KB) (PCKCACHESZ) = (MAXAPPLS*8)	First active log file =
Statistics heap size (4KB) (STAT_HEAP_SZ) = 4384	Block log on disk full (BLK_LOG_DSK_FUL) = NO
Interval for checking deadlock (ms) (DLCHKTIME) = 10000	Percent of max active log space by transaction(MAX_LOG) = 0
Percent. of lock lists per application (MAXLOCKS) = 20	Num. of active log files for 1 active UOW(NUM_LOG_SPAN) = 0
Lock timeout (sec) (LOCKTIMEOUT) = -1	Group commit count (MINCOMMIT) = 2
Changed pages threshold (CHNGPGS_THRESH) = 60	Percent log file reclaimed before soft ckcpt (SOFTMAX) = 360
Number of asynchronous page cleaners (NUM_IOCLEANERS) = 4	Log retain for recovery enabled (LOGRETAIN) = OFF
Number of I/O servers (NUM_IOSERVERS) = 4	User exit for logging enabled (USEREXIT) = OFF
Index sort flag (INDEXSORT) = YES	Auto restart enabled (AUTORESTART) = ON
Sequential detect flag (SEQDETECT) = YES	Index re-creation time (INDEXREC) = SYSTEM (RESTART)
Default prefetch size (pages) (DFT_PREFETCH_SZ) = 32	Default number of loadrec sessions (DFT_LOADREC_SES) = 1
Track modified pages (TRACKMOD) = OFF	Number of database backups to retain (NUM_DB_BACKUPS) = 12
Default number of containers = 1	Recovery history retention (days) (REC_HIS_RETENTN) = 366
Default tablespace extentsize (pages) (DFT_EXTENT_SZ) = 32	TSM management class (TSM_MGMTCLASS) =
Max number of active applications (MAXAPPLS) = AUTOMATIC	TSM node name (TSM_NODENAME) =
Average number of active applications (AVG_APPLS) = 1	TSM owner (TSM_OWNER) =
	TSM password (TSM_PASSWORD) =

Database Configuration for Node 3

Database Configuration for Database TPCD		Database heap (4KB) (DBHEAP)
Database configuration release level	=	= 10000
0x0a00		Size of database shared memory (4KB)
Database release level	=	(DATABASE_MEMORY) = AUTOMATIC
0x0a00		Catalog cache size (4KB)
Database territory	= US	(CATALOGCACHE_SZ) = (MAXAPPLS*4)
Database code page	= 819	Log buffer size (4KB) (LOGBUFSZ)
Database code set	=	= 2048
ISO8859-1		Utilities heap size (4KB)
Database country/region code	= 1	(UTIL_HEAP_SZ) = 5000
Dynamic SQL Query management		Buffer pool size (pages) (BUFFPAGE)
(DYN_QUERY_MGMT) = DISABLE		= 1000
Discovery support for this database		Extended storage segments size (4KB)
(DISCOVER_DB) = ENABLE		(ESTORE_SEG_SZ) = 16000
Default query optimization class		Number of extended storage segments
(DFT_QUERYOPT) = 7		(NUM_ESTORE_SEGS) = 0
Degree of parallelism		Max storage for lock list (4KB)
(DFT_DEGREE) = 1		(LOCKLIST) = 40000
Continue upon arithmetic exceptions		Max size of appl. group mem set (4KB)
(DFT_SQLMATHWARN) = NO		(APPGROUP_MEM_SZ) = 2000
Default refresh age		Percent of mem for appl. group heap
(DFT_REFRESH_AGE) = 0		(GROUPHEAP_RATIO) = 70
Number of frequent values retained		Max appl. control heap size (4KB)
(NUM_FREQVALUES) = 0		(APP_CTL_HEAP_SZ) = 512
Number of quantiles retained		Sort heap thres for shared sorts (4KB)
(NUM_QUANTILES) = 300		(SHEAPTHRES_SHR) = 250
Backup pending	= NO	Sort list heap (4KB) (SORTHEAP) =
Database is consistent	= YES	20000
Rollforward pending	= NO	SQL statement heap (4KB)
Restore pending	= NO	(STMTHEAP) = 10000
Multi-page file allocation enabled	=	Default application heap (4KB)
NO		(APPLHEAPSZ) = 1024
Log retain for recovery status	= NO	Package cache size (4KB)
User exit for logging status	= NO	(PCKCACHESZ) = (MAXAPPLS*8)
Data Links Token Expiry Interval (sec)		Statistics heap size (4KB)
(DL_EXPINT) = 60		(STAT_HEAP_SZ) = 4384
Data Links Write Token Init Expiry		Interval for checking deadlock (ms)
Intvl(DL_WT_IEXPINT) = 60		(DLCHKTIME) = 10000
Data Links Number of Copies		Percent. of lock lists per application
(DL_NUM_COPIES) = 1		(MAXLOCKS) = 20
Data Links Time after Drop (days)		Lock timeout (sec)
(DL_TIME_DROP) = 1		(LOCKTIMEOUT) = -1
Data Links Token in Uppercase		Changed pages threshold
(DL_UPPER) = NO		(CHNGPGS_THRESH) = 60
Data Links Token Algorithm		Number of asynchronous page cleaners
(DL_TOKEN) = MAC0		(NUM_IOCLEANERS) = 4
		Number of I/O servers
		(NUM_IOSERVERS) = 4
		Index sort flag (INDEXSORT) =
		YES
		Sequential detect flag (SEQDETECT)
		= YES

Default prefetch size (pages) (DFT_PREFETCH_SZ) = 32		Recovery history retention (days) (REC_HIS_RETENTN) = 366
Track modified pages (TRACKMOD) = OFF		TSM management class (TSM_MGMTCLASS) =
Default number of containers Default tablespace extentsize (pages) (DFT_EXTENT_SZ) = 32	= 1	TSM node name (TSM_NODENAME) =
Max number of active applications (MAXAPPLS) = AUTOMATIC		TSM owner (TSM_OWNER)
Average number of active applications (AVG_APPLS) = 1		=
Max DB files open per application (MAXFILOP) = 1024		TSM password (TSM_PASSWORD) =
		Database Configuration for Node 4
		Database Configuration for Database TPCD
Log file size (4KB) (LOGFILSIZ) = 50000		Database configuration release level = 0x0a00
Number of primary log files (LOGPRIMARY) = 4		Database release level = 0x0a00
Number of secondary log files (LOGSECOND) = 1		Database territory = US
Changed path to log files (NEWLOGPATH) =		Database code page = 819
Path to log files = /dev/raw/raw62	=	Database code set = ISO8859-1
Overflow log path (OVERFLOWLOGPATH) =		Database country/region code = 1
Mirror log path (MIRRORLOGPATH) =		Dynamic SQL Query management (DYN_QUERY_MGMT) = DISABLE
First active log file =	=	Discovery support for this database (DISCOVER_DB) = ENABLE
Block log on disk full (BLK_LOG_DSK_FUL) = NO		Default query optimization class (DFT_QUERYOPT) = 7
Percent of max active log space by transaction (MAX_LOG) = 0		Degree of parallelism (DFT_DEGREE) = 1
Num. of active log files for 1 active UOW (NUM_LOG_SPAN) = 0		Continue upon arithmetic exceptions (DFT_SQLMATHWARN) = NO
Group commit count (MINCOMMIT) = 2		Default refresh age (DFT_REFRESH_AGE) = 0
Percent log file reclaimed before soft ckcpt (SOFTMAX) = 360		Number of frequent values retained (NUM_FREQVALUES) = 0
Log retain for recovery enabled (LOGRETAIN) = OFF		Number of quantiles retained (NUM_QUANTILES) = 300
User exit for logging enabled (USEREXIT) = OFF		Backup pending = NO
Auto restart enabled (AUTORESTART) = ON		Database is consistent = YES
Index re-creation time (INDEXREC) = SYSTEM (RESTART)		Rollforward pending = NO
Default number of loadrec sessions (DFT_LOADREC_SES) = 1		Restore pending = NO
Number of database backups to retain (NUM_DB_BACKUPS) = 12		Multi-page file allocation enabled = NO

Log retain for recovery status	= NO	Percent. of lock lists per application
User exit for logging status	= NO	(MAXLOCKS) = 20
		Lock timeout (sec)
Data Links Token Expiry Interval (sec)		(LOCKTIMEOUT) = -1
(DL_EXPINT) = 60		
Data Links Write Token Init Expiry		Changed pages threshold
Intvl(DL_WT_IEXPINT) = 60		(CHNGPGS_THRESH) = 60
Data Links Number of Copies		Number of asynchronous page cleaners
(DL_NUM_COPIES) = 1		(NUM_IOCLEANERS) = 4
Data Links Time after Drop (days)		Number of I/O servers
(DL_TIME_DROP) = 1		(NUM_IOSERVERS) = 4
Data Links Token in Uppercase		Index sort flag
(DL_UPPER) = NO		(INDEXSORT) =
Data Links Token Algorithm		Sequential detect flag
(DL_TOKEN) = MAC0		= YES
		(SEQDETECT)
		Default prefetch size (pages)
Database heap (4KB)	(DBHEAP)	(DFT_PREFETCH_SZ) = 32
= 10000		
Size of database shared memory (4KB)		Track modified pages
(DATABASE_MEMORY) = AUTOMATIC		(TRACKMOD) = OFF
Catalog cache size (4KB)		
(CATALOGCACHE_SZ) = (MAXAPPLS*4)		Default number of containers
Log buffer size (4KB)	(LOGBUFSZ)	= 1
= 2048		Default tablespace extentsize (pages)
Utilities heap size (4KB)		(DFT_EXTENT_SZ) = 32
(UTIL_HEAP_SZ) = 5000		
Buffer pool size (pages)	(BUFFPAGE)	Max number of active applications
= 1000		(MAXAPPLS) = AUTOMATIC
Extended storage segments size (4KB)		Average number of active applications
(ESTORE_SEG_SZ) = 16000		(AVG_APPLS) = 1
Number of extended storage segments		Max DB files open per application
(NUM_ESTORE_SEGS) = 0		(MAXFILOP) = 1024
Max storage for lock list (4KB)		
(LOCKLIST) = 40000		Log file size (4KB)
		(LOGFILSIZ) =
		50000
Max size of appl. group mem set (4KB)		Number of primary log files
(APPGROUP_MEM_SZ) = 2000		(LOGPRIMARY) = 4
Percent of mem for appl. group heap		Number of secondary log files
(GROUPHEAP_RATIO) = 70		(LOGSECOND) = 1
Max appl. control heap size (4KB)		Changed path to log files
(APP_CTL_HEAP_SZ) = 512		(NEWLOGPATH) =
		Path to log files
		=
		/dev/raw/raw61
Sort heap thres for shared sorts (4KB)		Overflow log path
(SHEAPTHRES_SHR) = 250		(OVERFLOWLOGPATH) =
Sort list heap (4KB)	(SORTHEAP) =	Mirror log path
20000		(MIRRORLOGPATH) =
SQL statement heap (4KB)		First active log file
(STMTHEAP) = 10000		=
Default application heap (4KB)		Block log on disk full
(APPLHEAPSZ) = 1024		(BLK_LOG_DSK_FUL) = NO
Package cache size (4KB)		Percent of max active log space by
(PCKCACHESZ) = (MAXAPPLS*8)		transaction(MAX_LOG) = 0
Statistics heap size (4KB)		Num. of active log files for 1 active
(STAT_HEAP_SZ) = 4384		UOW(NUM_LOG_SPAN) = 0
Interval for checking deadlock (ms)		
(DLCHKTIME) = 10000		Group commit count
		(MINCOMMIT) = 2

Percent log file reclaimed before soft ckcpt
(SOFTMAX) = 360
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
Number of database backups to retain
(NUM_DB_BACKUPS) = 12
Recovery history retention (days)
(REC_HIS_RETENTN) = 366

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

Database Configuration for Node 5

Database Configuration for Database TPCD

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

Database territory = US
Database code page = 819
Database code set =
ISO8859-1
Database country/region code = 1

Dynamic SQL Query management
(DYN_QUERY_MGMT) = DISABLE

Discovery support for this database
(DISCOVER_DB) = ENABLE
Default query optimization class
(DFT_QUERYOPT) = 7
Degree of parallelism
(DFT_DEGREE) = 1
Continue upon arithmetic exceptions
(DFT_SQLMATHWARN) = NO
Default refresh age
(DFT_REFRESH_AGE) = 0

Number of frequent values retained
(NUM_FREQVALUES) = 0
Number of quantiles retained
(NUM_QUANTILES) = 300

Backup pending = NO

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

Multi-page file allocation enabled =
NO

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

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

Database heap (4KB) (DBHEAP)
= 10000
Size of database shared memory (4KB)
(DATABASE_MEMORY) = AUTOMATIC
Catalog cache size (4KB)
(CATALOGCACHE_SZ) = (MAXAPPLS*4)
Log buffer size (4KB) (LOGBUFSZ)
= 2048
Utilities heap size (4KB)
(UTIL_HEAP_SZ) = 5000
Buffer pool size (pages) (BUFFPAGE)
= 1000
Extended storage segments size (4KB)
(ESTORE_SEG_SZ) = 16000
Number of extended storage segments
(NUM_ESTORE_SEGS) = 0
Max storage for lock list (4KB)
(LOCKLIST) = 40000

Max size of appl. group mem set (4KB)
(APPGROUP_MEM_SZ) = 2000
Percent of mem for appl. group heap
(GROUPHEAP_RATIO) = 70
Max appl. control heap size (4KB)
(APP_CTL_HEAP_SZ) = 512

Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = 250		Overflow log path (OVERFLOWLOGPATH) =	
Sort list heap (4KB) 20000	(SORTHEAP) =	Mirror log path (MIRRORLOGPATH) =	
SQL statement heap (4KB) (STMTHEAP) = 10000		First active log file	=
Default application heap (4KB) (APPLHEAPSZ) = 1024		Block log on disk full (BLK_LOG_DSK_FUL) = NO	
Package cache size (4KB) (PCKCACHESZ) = (MAXAPPLS*8)		Percent of max active log space by transaction(MAX_LOG) = 0	
Statistics heap size (4KB) (STAT_HEAP_SZ) = 4384		Num. of active log files for 1 active UOW(NUM_LOG_SPAN) = 0	
Interval for checking deadlock (ms) (DLCHKTIME) = 10000		Group commit count (MINCOMMIT) = 2	
Percent. of lock lists per application (MAXLOCKS) = 20		Percent log file reclaimed before soft ckcpt (SOFTMAX) = 360	
Lock timeout (sec) (LOCKTIMEOUT) = -1		Log retain for recovery enabled (LOGRETAIN) = OFF	
Changed pages threshold (CHNGPGS_THRESH) = 60		User exit for logging enabled (USEREXIT) = OFF	
Number of asynchronous page cleaners (NUM_IOCLEANERS) = 4		Auto restart enabled (AUTORESTART) = ON	
Number of I/O servers (NUM_IOSERVERS) = 4		Index re-creation time (INDEXREC)	
Index sort flag YES	(INDEXSORT) =	= SYSTEM (RESTART)	
Sequential detect flag = YES	(SEQDETECT)	Default number of loadrec sessions (DFT_LOADREC_SES) = 1	
Default prefetch size (pages) (DFT_PREFETCH_SZ) = 32		Number of database backups to retain (NUM_DB_BACKUPS) = 12	
Track modified pages (TRACKMOD) = OFF		Recovery history retention (days) (REC_HIS_RETENTN) = 366	
Default number of containers = 1		TSM management class (TSM_MGMTCLASS) =	
Default tablespace extentsize (pages) (DFT_EXTENT_SZ) = 32		TSM node name (TSM_NODENAME) =	
Max number of active applications (MAXAPPLS) = AUTOMATIC		TSM owner (TSM_OWNER)	
Average number of active applications (AVG_APPLS) = 1		=	
Max DB files open per application (MAXFILOP) = 1024		TSM password (TSM_PASSWORD) =	
Log file size (4KB) 50000	(LOGFILSIZ) =	Database Configuration for Node 6	
Number of primary log files (LOGPRIMARY) = 4		Database Configuration for Database TPCD	
Number of secondary log files (LOGSECOND) = 1		Database configuration release level	=
Changed path to log files (NEWLOGPATH) =		0x0a00	
Path to log files /dev/raw/raw62	=	Database release level	=
		0x0a00	
		Database territory	= US
		Database code page	= 819
		Database code set	=
		ISO8859-1	

Database country/region code	= 1	Buffer pool size (pages) (BUFFPAGE)
Dynamic SQL Query management (DYN_QUERY_MGMT) = DISABLE		= 1000
Discovery support for this database (DISCOVER_DB) = ENABLE		Extended storage segments size (4KB) (ESTORE_SEG_SZ) = 16000
Default query optimization class (DFT_QUERYOPT) = 7		Number of extended storage segments (NUM_ESTORE_SEGS) = 0
Degree of parallelism (DFT_DEGREE) = 1		Max storage for lock list (4KB) (LOCKLIST) = 40000
Continue upon arithmetic exceptions (DFT_SQLMATHWARN) = NO		Max size of appl. group mem set (4KB) (APPGROUP_MEM_SZ) = 2000
Default refresh age (DFT_REFRESH_AGE) = 0		Percent of mem for appl. group heap (GROUPHEAP_RATIO) = 70
Number of frequent values retained (NUM_FREQVALUES) = 0		Max appl. control heap size (4KB) (APP_CTL_HEAP_SZ) = 512
Number of quantiles retained (NUM_QUANTILES) = 300		Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = 250
Backup pending	= NO	Sort list heap (4KB) (SORTHEAP) = 20000
Database is consistent	= YES	SQL statement heap (4KB) (STMTHEAP) = 10000
Rollforward pending	= NO	Default application heap (4KB) (APPLHEAPSZ) = 1024
Restore pending	= NO	Package cache size (4KB) (PCKCACHESZ) = (MAXAPPLS*8)
Multi-page file allocation enabled	=	Statistics heap size (4KB) (STAT_HEAP_SZ) = 4384
NO		Interval for checking deadlock (ms) (DLCHKTIME) = 10000
Log retain for recovery status	= NO	Percent. of lock lists per application (MAXLOCKS) = 20
User exit for logging status	= NO	Lock timeout (sec) (LOCKTIMEOUT) = -1
Data Links Token Expiry Interval (sec) (DL_EXPINT) = 60		Changed pages threshold (CHNGPGS_THRESH) = 60
Data Links Write Token Init Expiry Intvl(DL_WT_IEXPINT) = 60		Number of asynchronous page cleaners (NUM_IOCLEANERS) = 4
Data Links Number of Copies (DL_NUM_COPIES) = 1		Number of I/O servers (NUM_IOSERVERS) = 4
Data Links Time after Drop (days) (DL_TIME_DROP) = 1		Index sort flag (INDEXSORT) = YES
Data Links Token in Uppercase (DL_UPPER) = NO		Sequential detect flag (SEQDETECT) = YES
Data Links Token Algorithm (DL_TOKEN) = MAC0		Default prefetch size (pages) (DFT_PREFETCH_SZ) = 32
Database heap (4KB) (DBHEAP) = 10000		Track modified pages (TRACKMOD) = OFF
Size of database shared memory (4KB) (DATABASE_MEMORY) = AUTOMATIC		Default number of containers = 1
Catalog cache size (4KB) (CATALOGCACHE_SZ) = (MAXAPPLS*4)		Default tablespace extentsize (pages) (DFT_EXTENT_SZ) = 32
Log buffer size (4KB) (LOGBUFSZ) = 2048		
Utilities heap size (4KB) (UTIL_HEAP_SZ) = 5000		

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

Log file size (4KB) (LOGFILSIZ) = 50000
 Number of primary log files
 (LOGPRIMARY) = 4
 Number of secondary log files
 (LOGSECOND) = 1
 Changed path to log files
 (NEWLOGPATH) =
 Path to log files = /dev/raw/raw61
 Overflow log path
 (OVERFLOWLOGPATH) =
 Mirror log path
 (MIRRORLOGPATH) =
 First active log file =
 Block log on disk full
 (BLK_LOG_DSK_FUL) = NO
 Percent of max active log space by
 transaction(MAX_LOG) = 0
 Num. of active log files for 1 active
 UOW(NUM_LOG_SPAN) = 0

Group commit count
 (MINCOMMIT) = 2
 Percent log file reclaimed before soft ckcpt
 (SOFTMAX) = 360
 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
 Number of database backups to retain
 (NUM_DB_BACKUPS) = 12
 Recovery history retention (days)
 (REC_HIS_RETENTN) = 366

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

TSM password
 (TSM_PASSWORD) =

Database Configuration for Node 7

Database Configuration for Database TPCD

Database configuration release level = 0x0a00
 Database release level = 0x0a00
 Database territory = US
 Database code page = 819
 Database code set = ISO8859-1
 Database country/region code = 1

Dynamic SQL Query management
 (DYN_QUERY_MGMT) = DISABLE

Discovery support for this database
 (DISCOVER_DB) = ENABLE
 Default query optimization class
 (DFT_QUERYOPT) = 7
 Degree of parallelism
 (DFT_DEGREE) = 1
 Continue upon arithmetic exceptions
 (DFT_SQLMATHWARN) = NO
 Default refresh age
 (DFT_REFRESH_AGE) = 0
 Number of frequent values retained
 (NUM_FREQVALUES) = 0
 Number of quantiles retained
 (NUM_QUANTILES) = 300

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

Multi-page file allocation enabled = NO

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

Data Links Token Expiry Interval (sec)
 (DL_EXPINT) = 60
 Data Links Write Token Init Expiry
 Intvl(DL_WT_IEXPINT) = 60
 Data Links Number of Copies
 (DL_NUM_COPIES) = 1

Data Links Time after Drop (days) (DL_TIME_DROP) = 1	Number of I/O servers (NUM_IOSERVERS) = 4
Data Links Token in Uppercase (DL_UPPER) = NO	Index sort flag (INDEXSORT) = YES
Data Links Token Algorithm (DL_TOKEN) = MAC0	Sequential detect flag (SEQDETECT) = YES
Database heap (4KB) (DBHEAP) = 10000	Default prefetch size (pages) (DFT_PREFETCH_SZ) = 32
Size of database shared memory (4KB) (DATABASE_MEMORY) = AUTOMATIC	Track modified pages (TRACKMOD) = OFF
Catalog cache size (4KB) (CATALOGCACHE_SZ) = (MAXAPPLS*4)	Default number of containers = 1
Log buffer size (4KB) (LOGBUFSZ) = 2048	Default tablespace extentsize (pages) (DFT_EXTENT_SZ) = 32
Utilities heap size (4KB) (UTIL_HEAP_SZ) = 5000	Max number of active applications (MAXAPPLS) = AUTOMATIC
Buffer pool size (pages) (BUFFPAGE) = 1000	Average number of active applications (AVG_APPLS) = 1
Extended storage segments size (4KB) (ESTORE_SEG_SZ) = 16000	Max DB files open per application (MAXFILOP) = 1024
Number of extended storage segments (NUM_ESTORE_SEGS) = 0	Log file size (4KB) (LOGFILSIZ) = 50000
Max storage for lock list (4KB) (LOCKLIST) = 40000	Number of primary log files (LOGPRIMARY) = 4
Max size of appl. group mem set (4KB) (APPGROUP_MEM_SZ) = 2000	Number of secondary log files (LOGSECOND) = 1
Percent of mem for appl. group heap (GROUPHEAP_RATIO) = 70	Changed path to log files (NEWLOGPATH) =
Max appl. control heap size (4KB) (APP_CTL_HEAP_SZ) = 512	Path to log files = /dev/raw/raw62
Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = 250	Overflow log path (OVERFLOWLOGPATH) =
Sort list heap (4KB) (SORTHEAP) = 20000	Mirror log path (MIRRORLOGPATH) =
SQL statement heap (4KB) (STMTHEAP) = 10000	First active log file =
Default application heap (4KB) (APPLHEAPSZ) = 1024	Block log on disk full (BLK_LOG_DSK_FUL) = NO
Package cache size (4KB) (PCKCACHESZ) = (MAXAPPLS*8)	Percent of max active log space by transaction(MAX_LOG) = 0
Statistics heap size (4KB) (STAT_HEAP_SZ) = 4384	Num. of active log files for 1 active UOW(NUM_LOG_SPAN) = 0
Interval for checking deadlock (ms) (DLCHKTIME) = 10000	Group commit count (MINCOMMIT) = 2
Percent. of lock lists per application (MAXLOCKS) = 20	Percent log file reclaimed before soft ckcpt (SOFTMAX) = 360
Lock timeout (sec) (LOCKTIMEOUT) = -1	Log retain for recovery enabled (LOGRETAIN) = OFF
Changed pages threshold (CHNGPGS_THRESH) = 60	User exit for logging enabled (USEREXIT) = OFF
Number of asynchronous page cleaners (NUM_IOCLEANERS) = 4	Auto restart enabled (AUTORESTART) = ON

Index re-creation time = SYSTEM (RESTART)	(INDEXREC)	Rollforward pending	= NO
Default number of loadrec sessions (DFT_LOADREC_SES) = 1		Restore pending	= NO
Number of database backups to retain (NUM_DB_BACKUPS) = 12		Multi-page file allocation enabled	=
Recovery history retention (days) (REC_HIS_RETENTN) = 366		NO	
		Log retain for recovery status	= NO
		User exit for logging status	= NO

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

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

Database Configuration for Node 8

Database Configuration for Database TPCD

Database configuration release level 0x0a00	=	Database heap (4KB) (DBHEAP)	= 10000
Database release level 0x0a00	=	Size of database shared memory (4KB) (DATABASE_MEMORY) = AUTOMATIC	
Database territory Database code page Database code set ISO8859-1	= US = 819 =	Catalog cache size (4KB) (CATALOGCACHE_SZ) = (MAXAPPLS*4)	
Database country/region code	= 1	Log buffer size (4KB) (LOGBUFSZ)	= 2048
Dynamic SQL Query management (DYN_QUERY_MGMT) = DISABLE		Utilities heap size (4KB) (UTIL_HEAP_SZ) = 5000	
Discovery support for this database (DISCOVER_DB) = ENABLE		Buffer pool size (pages) (BUFFPAGE)	= 1000
Default query optimization class (DFT_QUERYOPT) = 7		Extended storage segments size (4KB) (ESTORE_SEG_SZ) = 16000	
Degree of parallelism (DFT_DEGREE) = 1		Number of extended storage segments (NUM_ESTORE_SEGS) = 0	
Continue upon arithmetic exceptions (DFT_SQLMATHWARN) = NO		Max storage for lock list (4KB) (LOCKLIST) = 40000	
Default refresh age (DFT_REFRESH_AGE) = 0		Max size of appl. group mem set (4KB) (APPGROUP_MEM_SZ) = 2000	
Number of frequent values retained (NUM_FREQVALUES) = 0		Percent of mem for appl. group heap (GROUPHEAP_RATIO) = 70	
Number of quantiles retained (NUM_QUANTILES) = 300		Max appl. control heap size (4KB) (APP_CTL_HEAP_SZ) = 512	
Backup pending	= NO	Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = 250	
Database is consistent	= YES	Sort list heap (4KB) (SORTHEAP)	= 20000
		SQL statement heap (4KB) (STMTHEAP) = 10000	
		Default application heap (4KB) (APPLHEAPSZ) = 1024	

Package cache size (4KB) (PCKCACHESZ) = (MAXAPPLS*8)		Percent of max active log space by transaction(MAX_LOG) = 0
Statistics heap size (4KB) (STAT_HEAP_SZ) = 4384		Num. of active log files for 1 active UOW(NUM_LOG_SPAN) = 0
Interval for checking deadlock (ms) (DLCHKTIME) = 10000		Group commit count (MINCOMMIT) = 2
Percent. of lock lists per application (MAXLOCKS) = 20		Percent log file reclaimed before soft ckcpt (SOFTMAX) = 360
Lock timeout (sec) (LOCKTIMEOUT) = -1		Log retain for recovery enabled (LOGRETAIN) = OFF
Changed pages threshold (CHNGPGS_THRESH) = 60		User exit for logging enabled (USEREXIT) = OFF
Number of asynchronous page cleaners (NUM_IOCLEANERS) = 4		Auto restart enabled (AUTORESTART) = ON
Number of I/O servers (NUM_IOSERVERS) = 4		Index re-creation time (INDEXREC) = SYSTEM (RESTART)
Index sort flag (INDEXSORT) = YES		Default number of loadrec sessions (DFT_LOADREC_SES) = 1
Sequential detect flag (SEQDETECT) = YES		Number of database backups to retain (NUM_DB_BACKUPS) = 12
Default prefetch size (pages) (DFT_PREFETCH_SZ) = 32		Recovery history retention (days) (REC_HIS_RETENTN) = 366
Track modified pages (TRACKMOD) = OFF		TSM management class (TSM_MGMTCLASS) =
Default number of containers = 1		TSM node name (TSM_NODENAME) =
Default tablespace extentsize (pages) (DFT_EXTENT_SZ) = 32		TSM owner (TSM_OWNER) =
Max number of active applications (MAXAPPLS) = AUTOMATIC		TSM password (TSM_PASSWORD) =
Average number of active applications (AVG_APPLS) = 1		
Max DB files open per application (MAXFILOP) = 1024		
Log file size (4KB) (LOGFILSIZ) = 50000		
Number of primary log files (LOGPRIMARY) = 4		Database configuration release level = 0x0a00
Number of secondary log files (LOGSECOND) = 1		Database release level = 0x0a00
Changed path to log files (NEWLOGPATH) =		Database territory = US
Path to log files = /dev/raw/raw61		Database code page = 819
Overflow log path (OVERFLOWLOGPATH) =		Database code set = ISO8859-1
Mirror log path (MIRRORLOGPATH) =		Database country/region code = 1
First active log file =		Dynamic SQL Query management (DYN_QUERY_MGMT) = DISABLE
Block log on disk full (BLK_LOG_DSK_FUL) = NO		Discovery support for this database (DISCOVER_DB) = ENABLE

Database Configuration for Node 9

Database Configuration for Database TPCD

Default query optimization class (DFT_QUERYOPT) = 7		Max size of appl. group mem set (4KB) (APPGROUP_MEM_SZ) = 2000
Degree of parallelism (DFT_DEGREE) = 1		Percent of mem for appl. group heap (GROUPHEAP_RATIO) = 70
Continue upon arithmetic exceptions (DFT_SQLMATHWARN) = NO		Max appl. control heap size (4KB) (APP_CTL_HEAP_SZ) = 512
Default refresh age (DFT_REFRESH_AGE) = 0		
Number of frequent values retained (NUM_FREQVALUES) = 0		Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = 250
Number of quantiles retained (NUM_QUANTILES) = 300		Sort list heap (4KB) (SORTHEAP) = 20000
Backup pending = NO		SQL statement heap (4KB) (STMTHEAP) = 10000
Database is consistent = YES		Default application heap (4KB) (APPLHEAPSZ) = 1024
Rollforward pending = NO		Package cache size (4KB)
Restore pending = NO		(PCKCACHESZ) = (MAXAPPLS*8)
Multi-page file allocation enabled =		Statistics heap size (4KB) (STAT_HEAP_SZ) = 4384
NO		Interval for checking deadlock (ms) (DLCHKTIME) = 10000
Log retain for recovery status = NO		Percent. of lock lists per application (MAXLOCKS) = 20
User exit for logging status = NO		Lock timeout (sec) (LOCKTIMEOUT) = -1
Data Links Token Expiry Interval (sec) (DL_EXPINT) = 60		
Data Links Write Token Init Expiry Intvl(DL_WT_IEXPINT) = 60		Changed pages threshold (CHNGPGS_THRESH) = 60
Data Links Number of Copies (DL_NUM_COPIES) = 1		Number of asynchronous page cleaners (NUM_IOCLEANERS) = 4
Data Links Time after Drop (days) (DL_TIME_DROP) = 1		Number of I/O servers (NUM_IOSERVERS) = 4
Data Links Token in Uppercase (DL_UPPER) = NO		Index sort flag (INDEXSORT) = YES
Data Links Token Algorithm (DL_TOKEN) = MAC0		Sequential detect flag (SEQDETECT) = YES
Database heap (4KB) (DBHEAP) = 10000		Default prefetch size (pages) (DFT_PREFETCH_SZ) = 32
Size of database shared memory (4KB) (DATABASE_MEMORY) = AUTOMATIC		Track modified pages (TRACKMOD) = OFF
Catalog cache size (4KB) (CATALOGCACHE_SZ) = (MAXAPPLS*4)		Default number of containers = 1
Log buffer size (4KB) (LOGBUFSZ) = 2048		Default tablespace extentsize (pages) (DFT_EXTENT_SZ) = 32
Utilities heap size (4KB) (UTIL_HEAP_SZ) = 5000		
Buffer pool size (pages) (BUFFPAGE) = 1000		Max number of active applications (MAXAPPLS) = AUTOMATIC
Extended storage segments size (4KB) (ESTORE_SEG_SZ) = 16000		Average number of active applications (AVG_APPLS) = 1
Number of extended storage segments (NUM_ESTORE_SEGS) = 0		Max DB files open per application (MAXFILOP) = 1024
Max storage for lock list (4KB) (LOCKLIST) = 40000		Log file size (4KB) (LOGFILSIZ) = 50000

Number of primary log files (LOGPRIMARY) = 4		Database configuration release level	=	0x0a00
Number of secondary log files (LOGSECOND) = 1		Database release level	=	0x0a00
Changed path to log files (NEWLOGPATH) =		Database territory	=	US
Path to log files	=	Database code page	=	819
/dev/raw/raw62		Database code set	=	ISO8859-1
Overflow log path (OVERFLOWLOGPATH) =		Database country/region code	=	1
Mirror log path (MIRRORLOGPATH) =		Dynamic SQL Query management (DYN_QUERY_MGMT) =		DISABLE
First active log file	=	Discovery support for this database (DISCOVER_DB) =		ENABLE
Block log on disk full (BLK_LOG_DSK_FUL) =		Default query optimization class (DFT_QUERYOPT) =		7
Percent of max active log space by transaction(MAX_LOG) =		Degree of parallelism (DFT_DEGREE) =		1
0		Continue upon arithmetic exceptions (DFT_SQLMATHWARN) =		NO
Num. of active log files for 1 active UOW(NUM_LOG_SPAN) =		Default refresh age (DFT_REFRESH_AGE) =		0
0		Number of frequent values retained (NUM_FREQVALUES) =		0
Group commit count (MINCOMMIT) =		Number of quantiles retained (NUM_QUANTILES) =		300
2		Backup pending	=	NO
Percent log file reclaimed before soft ckcpt (SOFTMAX) =		Database is consistent	=	YES
360		Rollforward pending	=	NO
Log retain for recovery enabled (LOGRETAIN) =		Restore pending	=	NO
OFF		Multi-page file allocation enabled	=	NO
User exit for logging enabled (USEREXIT) =		NO		
OFF		Log retain for recovery status	=	NO
Auto restart enabled (AUTORESTART) =		User exit for logging status	=	NO
ON		Data Links Token Expiry Interval (sec) (DL_EXPINT) =		60
Index re-creation time (INDEXREC) = SYSTEM (RESTART)		Data Links Write Token Init Expiry Intvl(DL_WT_IEXPINT) =		60
(INDEXREC)		Data Links Number of Copies (DL_NUM_COPIES) =		1
Default number of loadrec sessions (DFT_LOADREC_SES) =		Data Links Time after Drop (days) (DL_TIME_DROP) =		1
1		Data Links Token in Uppercase (DL_UPPER) =		NO
Number of database backups to retain (NUM_DB_BACKUPS) =		Data Links Token Algorithm (DL_TOKEN) =		MAC0
12		Database heap (4KB) (DBHEAP)	=	10000
Recovery history retention (days) (REC_HIS_RETENTN) =				
366				
TSM management class (TSM_MGMTCLASS) =				
TSM node name (TSM_NODENAME) =				
TSM owner (TSM_OWNER)	=			
(TSM_OWNER)				
TSM password (TSM_PASSWORD) =				
Database Configuration for Node 10				
Database Configuration for Database TPCD				

Size of database shared memory (4KB) (DATABASE_MEMORY) = AUTOMATIC	Track modified pages (TRACKMOD) = OFF
Catalog cache size (4KB) (CATALOGCACHE_SZ) = (MAXAPPLS*4)	Default number of containers = 1
Log buffer size (4KB) (LOGBUFSZ) = 2048	Default tablespace extentsize (pages) (DFT_EXTENT_SZ) = 32
Utilities heap size (4KB) (UTIL_HEAP_SZ) = 5000	Max number of active applications (MAXAPPLS) = AUTOMATIC
Buffer pool size (pages) (BUFFPAGE) = 1000	Average number of active applications (AVG_APPLS) = 1
Extended storage segments size (4KB) (ESTORE_SEG_SZ) = 16000	Max DB files open per application (MAXFILOP) = 1024
Number of extended storage segments (NUM_ESTORE_SEGS) = 0	Log file size (4KB) (LOGFILSIZ) = 50000
Max storage for lock list (4KB) (LOCKLIST) = 40000	Number of primary log files (LOGPRIMARY) = 4
Max size of appl. group mem set (4KB) (APPGROUP_MEM_SZ) = 2000	Number of secondary log files (LOGSECOND) = 1
Percent of mem for appl. group heap (GROUPHEAP_RATIO) = 70	Changed path to log files (NEWLOGPATH) =
Max appl. control heap size (4KB) (APP_CTL_HEAP_SZ) = 512	Path to log files = /dev/raw/raw61
Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = 250	Overflow log path (OVERFLOWLOGPATH) =
Sort list heap (4KB) (SORTHEAP) = 20000	Mirror log path (MIRRORLOGPATH) =
SQL statement heap (4KB) (STMTHEAP) = 10000	First active log file =
Default application heap (4KB) (APPLHEAPSZ) = 1024	Block log on disk full (BLK_LOG_DSK_FUL) = NO
Package cache size (4KB) (PCKCACHESZ) = (MAXAPPLS*8)	Percent of max active log space by transaction(MAX_LOG) = 0
Statistics heap size (4KB) (STAT_HEAP_SZ) = 4384	Num. of active log files for 1 active UOW(NUM_LOG_SPAN) = 0
Interval for checking deadlock (ms) (DLCHKTIME) = 10000	Group commit count (MINCOMMIT) = 2
Percent. of lock lists per application (MAXLOCKS) = 20	Percent log file reclaimed before soft ckcpt (SOFTMAX) = 360
Lock timeout (sec) (LOCKTIMEOUT) = -1	Log retain for recovery enabled (LOGRETAIN) = OFF
Changed pages threshold (CHNGPGS_THRESH) = 60	User exit for logging enabled (USEREXIT) = OFF
Number of asynchronous page cleaners (NUM_IOCLEANERS) = 4	Auto restart enabled (AUTORESTART) = ON
Number of I/O servers (NUM_IOSERVERS) = 4	Index re-creation time (INDEXREC) = SYSTEM (RESTART)
Index sort flag (INDEXSORT) = YES	Default number of loadrec sessions (DFT_LOADREC_SES) = 1
Sequential detect flag (SEQDETECT) = YES	Number of database backups to retain (NUM_DB_BACKUPS) = 12
Default prefetch size (pages) (DFT_PREFETCH_SZ) = 32	Recovery history retention (days) (REC_HIS_RETENTN) = 366

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

Database Configuration for Node 11

Database Configuration for Database TPCD

Database configuration release level = 0x0a00
Database release level = 0x0a00
Database territory = US
Database code page = 819
Database code set = ISO8859-1
Database country/region code = 1
Dynamic SQL Query management
(DYN_QUERY_MGMT) = DISABLE
Discovery support for this database
(DISCOVER_DB) = ENABLE
Default query optimization class
(DFT_QUERYOPT) = 7
Degree of parallelism
(DFT_DEGREE) = 1
Continue upon arithmetic exceptions
(DFT_SQLMATHWARN) = NO
Default refresh age
(DFT_REFRESH_AGE) = 0
Number of frequent values retained
(NUM_FREQVALUES) = 0
Number of quantiles retained
(NUM_QUANTILES) = 300
Backup pending = NO
Database is consistent = YES
Rollforward pending = NO
Restore pending = NO
Multi-page file allocation enabled = NO
Log retain for recovery status = NO
User exit for logging status = NO

Data Links Token Expiry Interval (sec)
(DL_EXPINT) = 60
Data Links Write Token Init Expiry
Intvl(DL_WT_IEXPINT) = 60
Data Links Number of Copies
(DL_NUM_COPIES) = 1
Data Links Time after Drop (days)
(DL_TIME_DROP) = 1
Data Links Token in Uppercase
(DL_UPPER) = NO
Data Links Token Algorithm
(DL_TOKEN) = MAC0
Database heap (4KB) (DBHEAP)
= 10000
Size of database shared memory (4KB)
(DATABASE_MEMORY) = AUTOMATIC
Catalog cache size (4KB)
(CATALOGCACHE_SZ) = (MAXAPPLS*4)
Log buffer size (4KB) (LOGBUFSZ)
= 2048
Utilities heap size (4KB)
(UTIL_HEAP_SZ) = 5000
Buffer pool size (pages) (BUFFPAGE)
= 1000
Extended storage segments size (4KB)
(ESTORE_SEG_SZ) = 16000
Number of extended storage segments
(NUM_ESTORE_SEGS) = 0
Max storage for lock list (4KB)
(LOCKLIST) = 40000
Max size of appl. group mem set (4KB)
(APPGROUP_MEM_SZ) = 2000
Percent of mem for appl. group heap
(GROUPHEAP_RATIO) = 70
Max appl. control heap size (4KB)
(APP_CTL_HEAP_SZ) = 512
Sort heap thres for shared sorts (4KB)
(SHEAPTHRES_SHR) = 250
Sort list heap (4KB) (SORTHEAP) =
20000
SQL statement heap (4KB)
(STMTHEAP) = 10000
Default application heap (4KB)
(APPLHEAPSZ) = 1024
Package cache size (4KB)
(PCKCACHESZ) = (MAXAPPLS*8)
Statistics heap size (4KB)
(STAT_HEAP_SZ) = 4384
Interval for checking deadlock (ms)
(DLCHKTIME) = 10000
Percent. of lock lists per application
(MAXLOCKS) = 20

Lock timeout (sec)
 (LOCKTIMEOUT) = -1

Changed pages threshold
 (CHNGPGS_THRESH) = 60

Number of asynchronous page cleaners
 (NUM_IOCLEANERS) = 4

Number of I/O servers
 (NUM_IOSERVERS) = 4

Index sort flag (INDEXSORT) = YES

Sequential detect flag (SEQDETECT) = YES

Default prefetch size (pages)
 (DFT_PREFETCH_SZ) = 32

Track modified pages
 (TRACKMOD) = OFF

Default number of containers = 1

Default tablespace extentsize (pages)
 (DFT_EXTENT_SZ) = 32

Max number of active applications
 (MAXAPPLS) = AUTOMATIC

Average number of active applications
 (AVG_APPLS) = 1

Max DB files open per application
 (MAXFILOP) = 1024

Log file size (4KB) (LOGFILSIZ) = 50000

Number of primary log files
 (LOGPRIMARY) = 4

Number of secondary log files
 (LOGSECOND) = 1

Changed path to log files
 (NEWLOGPATH) =

Path to log files = /dev/raw/raw62

Overflow log path
 (OVERFLOWLOGPATH) =

Mirror log path
 (MIRRORLOGPATH) =

First active log file =

Block log on disk full
 (BLK_LOG_DSK_FUL) = NO

Percent of max active log space by transaction
 (MAX_LOG) = 0

Num. of active log files for 1 active UOW
 (NUM_LOG_SPAN) = 0

Group commit count
 (MINCOMMIT) = 2

Percent log file reclaimed before soft ckcpt
 (SOFTMAX) = 360

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

Number of database backups to retain
 (NUM_DB_BACKUPS) = 12

Recovery history retention (days)
 (REC_HIS_RETENTN) = 366

TSM management class
 (TSM_MGMTCLASS) =

TSM node name
 (TSM_NODENAME) =

TSM owner (TSM_OWNER) =

TSM password
 (TSM_PASSWORD) =

Database Configuration for Node 12

Database Configuration for Database TPCD

Database configuration release level = 0x0a00

Database release level = 0x0a00

Database territory = US

Database code page = 819

Database code set = ISO8859-1

Database country/region code = 1

Dynamic SQL Query management
 (DYN_QUERY_MGMT) = DISABLE

Discovery support for this database
 (DISCOVER_DB) = ENABLE

Default query optimization class
 (DFT_QUERYOPT) = 7

Degree of parallelism
 (DFT_DEGREE) = 1

Continue upon arithmetic exceptions
 (DFT_SQLMATHWARN) = NO

Default refresh age
 (DFT_REFRESH_AGE) = 0

Number of frequent values retained (NUM_FREQVALUES) = 0		Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = 250
Number of quantiles retained (NUM_QUANTILES) = 300		Sort list heap (4KB) (SORTHEAP) = 20000
Backup pending = NO		SQL statement heap (4KB) (STMTHEAP) = 10000
Database is consistent = YES		Default application heap (4KB) (APPLHEAPSZ) = 1024
Rollforward pending = NO		Package cache size (4KB) (PCKCACHESZ) = (MAXAPPLS*8)
Restore pending = NO		Statistics heap size (4KB) (STAT_HEAP_SZ) = 4384
Multi-page file allocation enabled = NO		Interval for checking deadlock (ms) (DLCHKTIME) = 10000
Log retain for recovery status = NO		Percent. of lock lists per application (MAXLOCKS) = 20
User exit for logging status = NO		Lock timeout (sec) (LOCKTIMEOUT) = -1
Data Links Token Expiry Interval (sec) (DL_EXPINT) = 60		Changed pages threshold (CHNGPGS_THRESH) = 60
Data Links Write Token Init Expiry Intvl(DL_WT_IEXPINT) = 60		Number of asynchronous page cleaners (NUM_IOCLEANERS) = 4
Data Links Number of Copies (DL_NUM_COPIES) = 1		Number of I/O servers (NUM_IOSERVERS) = 4
Data Links Time after Drop (days) (DL_TIME_DROP) = 1		Index sort flag (INDEXSORT) = YES
Data Links Token in Uppercase (DL_UPPER) = NO		Sequential detect flag (SEQDETECT) = YES
Data Links Token Algorithm (DL_TOKEN) = MAC0		Default prefetch size (pages) (DFT_PREFETCH_SZ) = 32
Database heap (4KB) (DBHEAP) = 10000		Track modified pages (TRACKMOD) = OFF
Size of database shared memory (4KB) (DATABASE_MEMORY) = AUTOMATIC		Default number of containers = 1
Catalog cache size (4KB) (CATALOGCACHE_SZ) = (MAXAPPLS*4)		Default tablespace extentsize (pages) (DFT_EXTENT_SZ) = 32
Log buffer size (4KB) (LOGBUFSZ) = 2048		Max number of active applications (MAXAPPLS) = AUTOMATIC
Utilities heap size (4KB) (UTIL_HEAP_SZ) = 5000		Average number of active applications (AVG_APPLS) = 1
Buffer pool size (pages) (BUFFPAGE) = 1000		Max DB files open per application (MAXFILOP) = 1024
Extended storage segments size (4KB) (ESTORE_SEG_SZ) = 16000		Log file size (4KB) (LOGFILSIZ) = 50000
Number of extended storage segments (NUM_ESTORE_SEGS) = 0		Number of primary log files (LOGPRIMARY) = 4
Max storage for lock list (4KB) (LOCKLIST) = 40000		Number of secondary log files (LOGSECOND) = 1
Max size of appl. group mem set (4KB) (APPGROUP_MEM_SZ) = 2000		Changed path to log files (NEWLOGPATH) =
Percent of mem for appl. group heap (GROUPHEAP_RATIO) = 70		Path to log files =
Max appl. control heap size (4KB) (APP_CTL_HEAP_SZ) = 512		/dev/raw/raw61

Overflow log path
 (OVERFLOWLOGPATH) =
 Mirror log path
 (MIRRORLOGPATH) =
 First active log file =
 Block log on disk full
 (BLK_LOG_DSK_FUL) = NO
 Percent of max active log space by
 transaction(MAX_LOG) = 0
 Num. of active log files for 1 active
 UOW(NUM_LOG_SPAN) = 0

 Group commit count
 (MINCOMMIT) = 2
 Percent log file reclaimed before soft ckcpt
 (SOFTMAX) = 360
 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
 Number of database backups to retain
 (NUM_DB_BACKUPS) = 12
 Recovery history retention (days)
 (REC_HIS_RETENTN) = 366

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

Database Configuration for Node 13

Database Configuration for Database TPCD

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

 Database territory = US
 Database code page = 819
 Database code set =
 ISO8859-1
 Database country/region code = 1

Dynamic SQL Query management
 (DYN_QUERY_MGMT) = DISABLE

 Discovery support for this database
 (DISCOVER_DB) = ENABLE
 Default query optimization class
 (DFT_QUERYOPT) = 7
 Degree of parallelism
 (DFT_DEGREE) = 1
 Continue upon arithmetic exceptions
 (DFT_SQLMATHWARN) = NO
 Default refresh age
 (DFT_REFRESH_AGE) = 0
 Number of frequent values retained
 (NUM_FREQVALUES) = 0
 Number of quantiles retained
 (NUM_QUANTILES) = 300

 Backup pending = NO

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

 Multi-page file allocation enabled =
 NO

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

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

 Database heap (4KB) (DBHEAP)
 = 10000
 Size of database shared memory (4KB)
 (DATABASE_MEMORY) = AUTOMATIC
 Catalog cache size (4KB)
 (CATALOGCACHE_SZ) = (MAXAPPLS*4)
 Log buffer size (4KB) (LOGBUFSZ)
 = 2048
 Utilities heap size (4KB)
 (UTIL_HEAP_SZ) = 5000
 Buffer pool size (pages) (BUFFPAGE)
 = 1000

Extended storage segments size (4KB) (ESTORE_SEG_SZ) = 16000	Average number of active applications (AVG_APPLS) = 1
Number of extended storage segments (NUM_ESTORE_SEGS) = 0	Max DB files open per application (MAXFILOP) = 1024
Max storage for lock list (4KB) (LOCKLIST) = 40000	Log file size (4KB) (LOGFILSIZ) = 50000
Max size of appl. group mem set (4KB) (APPGROUP_MEM_SZ) = 2000	Number of primary log files (LOGPRIMARY) = 4
Percent of mem for appl. group heap (GROUPHEAP_RATIO) = 70	Number of secondary log files (LOGSECOND) = 1
Max appl. control heap size (4KB) (APP_CTL_HEAP_SZ) = 512	Changed path to log files (NEWLOGPATH) =
Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = 250	Path to log files = /dev/raw/raw62
Sort list heap (4KB) (SORTHEAP) = 20000	Overflow log path (OVERFLOWLOGPATH) =
SQL statement heap (4KB) (STMTHEAP) = 10000	Mirror log path (MIRRORLOGPATH) =
Default application heap (4KB) (APPLHEAPSZ) = 1024	First active log file =
Package cache size (4KB) (PCKCACHESZ) = (MAXAPPLS*8)	Block log on disk full (BLK_LOG_DSK_FUL) = NO
Statistics heap size (4KB) (STAT_HEAP_SZ) = 4384	Percent of max active log space by transaction(MAX_LOG) = 0
Interval for checking deadlock (ms) (DLCHKTIME) = 10000	Num. of active log files for 1 active UOW(NUM_LOG_SPAN) = 0
Percent. of lock lists per application (MAXLOCKS) = 20	Group commit count (MINCOMMIT) = 2
Lock timeout (sec) (LOCKTIMEOUT) = -1	Percent log file reclaimed before soft ckcpt (SOFTMAX) = 360
Changed pages threshold (CHNGPGS_THRESH) = 60	Log retain for recovery enabled (LOGRETAIN) = OFF
Number of asynchronous page cleaners (NUM_IOCLEANERS) = 4	User exit for logging enabled (USEREXIT) = OFF
Number of I/O servers (NUM_IOSERVERS) = 4	Auto restart enabled (AUTORESTART) = ON
Index sort flag (INDEXSORT) = YES	Index re-creation time (INDEXREC) = SYSTEM (RESTART)
Sequential detect flag (SEQDETECT) = YES	Default number of loadrec sessions (DFT_LOADREC_SES) = 1
Default prefetch size (pages) (DFT_PREFETCH_SZ) = 32	Number of database backups to retain (NUM_DB_BACKUPS) = 12
Track modified pages (TRACKMOD) = OFF	Recovery history retention (days) (REC_HIS_RETENTN) = 366
Default number of containers = 1	TSM management class (TSM_MGMTCLASS) =
Default tablespace extentsize (pages) (DFT_EXTENT_SZ) = 32	TSM node name (TSM_NODENAME) =
Max number of active applications (MAXAPPLS) = AUTOMATIC	TSM owner (TSM_OWNER) = TSM password (TSM_PASSWORD) =

Database Configuration for Node 14

Database Configuration for Database TPCD		Data Links Token in Uppercase (DL_UPPER) = NO
		Data Links Token Algorithm (DL_TOKEN) = MAC0
Database configuration release level 0x0a00	=	Database heap (4KB) (DBHEAP) = 10000
Database release level 0x0a00	=	Size of database shared memory (4KB) (DATABASE_MEMORY) = AUTOMATIC
Database territory	= US	Catalog cache size (4KB) (CATALOGCACHE_SZ) = (MAXAPPLS*4)
Database code page	= 819	Log buffer size (4KB) (LOGBUFSZ)
Database code set ISO8859-1	=	= 2048
Database country/region code	= 1	Utilities heap size (4KB) (UTIL_HEAP_SZ) = 5000
Dynamic SQL Query management (DYN_QUERY_MGMT) = DISABLE		Buffer pool size (pages) (BUFFPAGE) = 1000
Discovery support for this database (DISCOVER_DB) = ENABLE		Extended storage segments size (4KB) (ESTORE_SEG_SZ) = 16000
Default query optimization class (DFT_QUERYOPT) = 7		Number of extended storage segments (NUM_ESTORE_SEGS) = 0
Degree of parallelism (DFT_DEGREE) = 1		Max storage for lock list (4KB) (LOCKLIST) = 40000
Continue upon arithmetic exceptions (DFT_SQLMATHWARN) = NO		Max size of appl. group mem set (4KB) (APPGROUP_MEM_SZ) = 2000
Default refresh age (DFT_REFRESH_AGE) = 0		Percent of mem for appl. group heap (GROUPHEAP_RATIO) = 70
Number of frequent values retained (NUM_FREQVALUES) = 0		Max appl. control heap size (4KB) (APP_CTL_HEAP_SZ) = 512
Number of quantiles retained (NUM_QUANTILES) = 300		Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = 250
Backup pending	= NO	Sort list heap (4KB) (SORTHEAP) = 20000
Database is consistent	= YES	SQL statement heap (4KB) (STMTHEAP) = 10000
Rollforward pending	= NO	Default application heap (4KB) (APPLHEAPSZ) = 1024
Restore pending	= NO	Package cache size (4KB) (PCKCACHESZ) = (MAXAPPLS*8)
Multi-page file allocation enabled NO	=	Statistics heap size (4KB) (STAT_HEAP_SZ) = 4384
Log retain for recovery status	= NO	Interval for checking deadlock (ms) (DLCHKTIME) = 10000
User exit for logging status	= NO	Percent. of lock lists per application (MAXLOCKS) = 20
Data Links Token Expiry Interval (sec) (DL_EXPINT) = 60		Lock timeout (sec) (LOCKTIMEOUT) = -1
Data Links Write Token Init Expiry Intvl(DL_WT_IEXPINT) = 60		Changed pages threshold (CHNGPGS_THRESH) = 60
Data Links Number of Copies (DL_NUM_COPIES) = 1		Number of asynchronous page cleaners (NUM_IOCLEANERS) = 4
Data Links Time after Drop (days) (DL_TIME_DROP) = 1		Number of I/O servers (NUM_IOSERVERS) = 4

Index sort flag YES	(INDEXSORT) =	Default number of loadrec sessions (DFT_LOADREC_SES) = 1
Sequential detect flag = YES	(SEQDETECT)	Number of database backups to retain (NUM_DB_BACKUPS) = 12
Default prefetch size (pages) (DFT_PREFETCH_SZ) = 32		Recovery history retention (days) (REC_HIS_RETENTN) = 366
Track modified pages (TRACKMOD) = OFF		TSM management class (TSM_MGMTCLASS) =
Default number of containers Default tablespace extentsize (pages) (DFT_EXTENT_SZ) = 32	= 1	TSM node name (TSM_NODENAME) = TSM owner (TSM_OWNER)
Max number of active applications (MAXAPPLS) = AUTOMATIC		TSM password (TSM_PASSWORD) =
Average number of active applications (AVG_APPLS) = 1		
Max DB files open per application (MAXFILOP) = 1024		
Log file size (4KB) 50000	(LOGFILSIZ) =	
Number of primary log files (LOGPRIMARY) = 4		
Number of secondary log files (LOGSECOND) = 1		
Changed path to log files (NEWLOGPATH) =		
Path to log files /dev/raw/raw61	=	
Overflow log path (OVERFLOWLOGPATH) =		
Mirror log path (MIRRORLOGPATH) =		
First active log file Block log on disk full (BLK_LOG_DSK_FUL) = NO	=	
Percent of max active log space by transaction(MAX_LOG) = 0		
Num. of active log files for 1 active UOW(NUM_LOG_SPAN) = 0		
Group commit count (MINCOMMIT) = 2		
Percent log file reclaimed before soft chckpt (SOFTMAX) = 360		
Log retain for recovery enabled (LOGRETAIN) = OFF		
User exit for logging enabled (USEREXIT) = OFF		
Auto restart enabled (AUTORESTART) = ON		
Index re-creation time = SYSTEM (RESTART)	(INDEXREC)	
		Database Configuration for Node 15
		Database Configuration for Database TPCD
		Database configuration release level = 0x0a00
		Database release level = 0x0a00
		Database territory = US
		Database code page = 819
		Database code set = ISO8859-1
		Database country/region code = 1
		Dynamic SQL Query management (DYN_QUERY_MGMT) = DISABLE
		Discovery support for this database (DISCOVER_DB) = ENABLE
		Default query optimization class (DFT_QUERYOPT) = 7
		Degree of parallelism (DFT_DEGREE) = 1
		Continue upon arithmetic exceptions (DFT_SQLMATHWARN) = NO
		Default refresh age (DFT_REFRESH_AGE) = 0
		Number of frequent values retained (NUM_FREQVALUES) = 0
		Number of quantiles retained (NUM_QUANTILES) = 300
		Backup pending = NO
		Database is consistent = YES
		Rollforward pending = NO
		Restore pending = NO

Multi-page file allocation enabled NO	=	Statistics heap size (4KB) (STAT_HEAP_SZ) = 4384
Log retain for recovery status	= NO	Interval for checking deadlock (ms) (DLCHKTIME) = 10000
User exit for logging status	= NO	Percent. of lock lists per application (MAXLOCKS) = 20
Data Links Token Expiry Interval (sec) (DL_EXPINT) = 60		Lock timeout (sec) (LOCKTIMEOUT) = -1
Data Links Write Token Init Expiry Intvl(DL_WT_IEXPINT) = 60		Changed pages threshold (CHNGPGS_THRESH) = 60
Data Links Number of Copies (DL_NUM_COPIES) = 1		Number of asynchronous page cleaners (NUM_IOCLEANERS) = 4
Data Links Time after Drop (days) (DL_TIME_DROP) = 1		Number of I/O servers (NUM_IOSERVERS) = 4
Data Links Token in Uppercase (DL_UPPER) = NO		Index sort flag (INDEXSORT) =
Data Links Token Algorithm (DL_TOKEN) = MAC0		YES
Database heap (4KB) (DBHEAP) = 10000		Sequential detect flag (SEQDETECT) = YES
Size of database shared memory (4KB) (DATABASE_MEMORY) = AUTOMATIC		Default prefetch size (pages) (DFT_PREFETCH_SZ) = 32
Catalog cache size (4KB) (CATALOGCACHE_SZ) = (MAXAPPLS*4)		Track modified pages (TRACKMOD) = OFF
Log buffer size (4KB) (LOGBUFSZ) = 2048		Default number of containers = 1
Utilities heap size (4KB) (UTIL_HEAP_SZ) = 5000		Default tablespace extentsize (pages) (DFT_EXTENT_SZ) = 32
Buffer pool size (pages) (BUFFPAGE) = 1000		Max number of active applications (MAXAPPLS) = AUTOMATIC
Extended storage segments size (4KB) (ESTORE_SEG_SZ) = 16000		Average number of active applications (AVG_APPLS) = 1
Number of extended storage segments (NUM_ESTORE_SEGS) = 0		Max DB files open per application (MAXFILOP) = 1024
Max storage for lock list (4KB) (LOCKLIST) = 40000		Log file size (4KB) (LOGFILSIZ) = 50000
Max size of appl. group mem set (4KB) (APPGROUP_MEM_SZ) = 2000		Number of primary log files (LOGPRIMARY) = 4
Percent of mem for appl. group heap (GROUPHEAP_RATIO) = 70		Number of secondary log files (LOGSECOND) = 1
Max appl. control heap size (4KB) (APP_CTL_HEAP_SZ) = 512		Changed path to log files (NEWLOGPATH) =
Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = 250		Path to log files =
Sort list heap (4KB) (SORTHEAP) = 20000		/dev/raw/raw62
SQL statement heap (4KB) (STMTHEAP) = 10000		Overflow log path (OVERFLOWLOGPATH) =
Default application heap (4KB) (APPLHEAPSZ) = 1024		Mirror log path (MIRRORLOGPATH) =
Package cache size (4KB) (PCKCACHESZ) = (MAXAPPLS*8)		First active log file =
		Block log on disk full (BLK_LOG_DSK_FUL) = NO
		Percent of max active log space by transaction(MAX_LOG) = 0

Num. of active log files for 1 active
 UOW(NUM_LOG_SPAN) = 0

Group commit count
 (MINCOMMIT) = 2
 Percent log file reclaimed before soft ckcpt
 (SOFTMAX) = 360
 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
 Number of database backups to retain
 (NUM_DB_BACKUPS) = 12
 Recovery history retention (days)
 (REC_HIS_RETENTN) = 366

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

DB2 Database Manager Configuration

Database Manager Configuration

Node type = Enterprise Server Edition with local
 and remote clients

Database manager configuration release level
 = 0x0a00

CPU speed (millisec/instruction)
 (CPUSPEED) = 2.715979e-07
 Communications bandwidth (MB/sec)
 (COMM_BANDWIDTH) = 1.000000e+00

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

Default charge-back account
 (DFT_ACCOUNT_STR) =

Java Development Kit installation path
 (JDK_PATH) = /opt/IBMJava2-amd64-131

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

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

SYSADM group name
 (SYSADM_GROUP) =
 SYSCTRL group name
 (SYSCTRL_GROUP) =
 SYSMANT group name
 (SYSMAINT_GROUP) =

Database manager authentication
 (AUTHENTICATION) = SERVER
 Cataloging allowed without authority
 (CATALOG_NOAUTH) = NO
 Trust all clients
 (TRUST_ALLCLNTS) = YES
 Trusted client authentication
 (TRUST_CLNTAUTH) = CLIENT
 Bypass federated authentication
 (FED_NOAUTH) = NO
 Default database path (DFTDBPATH)
 = /home/tpch

Database monitor heap size (4KB)
 (MON_HEAP_SZ) = 90
 Java Virtual Machine heap size (4KB)
 (JAVA_HEAP_SZ) = 2048

Audit buffer size (4KB)
 (AUDIT_BUF_SZ) = 0
 Size of instance shared memory (4KB)
 (INSTANCE_MEMORY) = AUTOMATIC
 Backup buffer default size (4KB)
 (BACKBUFSZ) = 1024
 Restore buffer default size (4KB)
 (RESTBUFSZ) = 1024

 Sort heap threshold (4KB)
 (SHEAPTHRES) = 500000

 Directory cache support
 (DIR_CACHE) = YES

 Application support layer heap size (4KB)
 (ASLHEAPSZ) = 15
 Max requester I/O block size (bytes)
 (RQRIOLBK) = 32767
 Query heap size (4KB)
 (QUERY_HEAP_SZ) = 1000
 DRDA services heap size (4KB)
 (DRDA_HEAP_SZ) = 128

 Workload impact by throttled
 utilities(UTIL_IMPACT_LIM) = 100

 Priority of agents (AGENTPRI) =
 SYSTEM
 Max number of existing agents
 (MAXAGENTS) = 400
 Agent pool size
 (NUM_POOLAGENTS) = 64
 Initial number of agents in pool
 (NUM_INITAGENTS) = 4
 Max number of coordinating agents
 (MAX_COORDAGENTS) = (MAXAGENTS -
 NUM_INITAGENTS)
 Max no. of concurrent coordinating agents
 (MAXCAGENTS) = MAX_COORDAGENTS
 Max number of client connections
 (MAX_CONNECTIONS) =
 MAX_COORDAGENTS

 Keep fenced process
 (KEEPFENCED) = YES
 Number of pooled fenced processes
 (FENCED_POOL) = MAX_COORDAGENTS
 Initialize fenced process with JVM
 (INITFENCED_JVM) = NO
 Initial number of fenced processes
 (NUM_INITFENCED) = 0

 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
 SPM log path
 (SPM_LOG_PATH) =

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

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

 No. of int. communication
 buffers(4KB)(FCM_NUM_BUFFERS) = 16384
 Node connection elapse time (sec)
 (CONN_ELAPSE) = 20
 Max number of node connection retries
 (MAX_CONNRETRIES) = 5
 Max time difference between nodes (min)
 (MAX_TIME_DIFF) = 1440

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

DB2 Registry Variables

DB2_EXTENDED_OPTIMIZATION=Y
 DB2_ANTIJOIN=Y
 DB2_LIKE_VARCHAR=Y,Y
 DB2BPVARS=/home/tpch/custom/bpvar.cfg
 DB2RQTIME=30
 DB2_FORCE_FCM_BP=ON
 DB2OPTIONS=-t -v +c
 DB2BQTRY=120
 DB2_PARALLEL_IO=*

Linux Parameters

```

echo "250 256000 100 1024" >
/proc/sys/kernel/sem
echo 262143 > /proc/sys/net/core/rmem_max
echo 262143 > /proc/sys/net/core/wmem_max
echo 262143 > /proc/sys/net/core/rmem_default
echo 262143 > /proc/sys/net/core/wmem_default

```

Appendix B: Database Build Scripts

buildtpcd

```

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

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

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

# Make output unbuffered.
open(STDOUT, "| tee buildtpcd.out.${timestamp}");
select(STDOUT);
$|= 1;
#-----#
# verify that necessary environment variables for
building the database #
# are present. Default those that aren't necessary
#
#-----#

# variables that must be specified for script to run
@reqVars = ("TPCD_PLATFORM",

```

```

"TPCD_PRODUCT",
"TPCD_VERSION",
"TPCD_DBNAME",
"TPCD_MODE",
"TPCD_SF",
"TPCD_DDLPATH",
"TPCD_AUDIT",
"TPCD_AUDIT_DIR",
"TPCD_BUILD_STAGE");

# variables default to 'NULL' if unspecified
@defNullVars = ("TPCD_LOAD_SCRIPT",
"TPCD_LOAD_SCRIPT_QUAL",
"TPCD_INPUT",
"TPCD_QUAL_INPUT",
"TPCD_DBGEN",
"TPCD_LOGPRIMARY",
"TPCD_LOGSECOND",
"TPCD_LOGFILSIZ",
"TPCD_LOG_DIR",
"TPCD_MACHINE",
"TPCD_AGENTPRI",

"TPCD_STAGING_TABLE_DDL",

"TPCD_PRELOAD_STAGING_TABLE_
SCRIPT",

"TPCD_DELETE_STAGING_TABLE_SQ
L",

"TPCD_RUNSTATSHORT",
"TPCD_ADD_RI",
"TPCD_AST",
"TPCD_DBM_CONFIG",
"TPCD_EXPLAIN_DDL",
"TPCD_NODEGROUP_DEF",
"TPCD_BUFFERPOOL_DEF",

"TPCD_LOAD_DB2SET_SCRIPT",
"TPCD_DB2SET_SCRIPT",

"TPCD_LOG_DIR_SETUP_SCRIPT",
"TPCD_LOAD_CONFIGFILE",

"TPCD_LOAD_DBM_CONFIGFILE",
"TPCD_TEMP");

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

if ( $qual eq "QUAL" ){
    @reqQualVars =
        ("TPCD_QUAL_DBNAME",
"TPCD_QUAL_DDL",

```

```

"TPCD_QUAL_TBSP_DDL",
"TPCD_QUALCONFIGFILE",
"TPCD_DBM_QUALCONFIG",
"TPCD_LOAD_QUALCONFIGFILE",
"TPCD_LOAD_DBM_QUALCONFIGFILE");

    &setVar(@reqQualVars, "ERROR");

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

$platform=$ENV{"TPCD_PLATFORM"};

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

# ddl script files found under custom directory
#
#-----#
-----#

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

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

if (length($ENV{"TPCD_BACKUP_DIR"}) <= 0){
    $ENV{"TPCD_BACKUP_DIR"} =
"${delim}dev${delim}null";
}
if (length($ENV{"TPCD_COPY_DIR"}) <= 0){
    $ENV{"TPCD_COPY_DIR"} =
"${delim}dev${delim}null";
}
if (length($ENV{"TPCD_TEMP"}) <= 1){
    $ENV{"TPCD_TEMP"} =
"/u/$instance/sqllib/tmp";
}
if (length($ENV{"TPCD_PHYS_NODE"}) <= 0){
    $ENV{"TPCD_NODEGROUP_DEF"}="NULL"
}
if
(length($ENV{"TPCD_GENERATE_SEED_FILE"
}) <= 0){
    $ENV{"TPCD_GENERATE_SEED_FILE"} =
"no";
}
if (length($ENV{"TPCD_SORTBUF"}) <= 0){
    $ENV{"TPCD_SORTBUF"} = 4096;
}
}
#-----#
-----#

```

```

if
(length($ENV{"TPCD_LOAD_PARALLELISM"})
<= 0){
    $ENV{"TPCD_LOAD_PARALLELISM"} = 0;
}
if (length($ENV{"TPCD_LOADSTATS"}) <= 0){
    $ENV{"TPCD_LOADSTATS"} = "no";
}
if (length($ENV{"TPCD_FASTPARSE"}) <= 0){
    $ENV{"TPCD_FASTPARSE"} = "no";
}
if (length($ENV{"TPCD_LOG"}) <= 0){
    $ENV{"TPCD_LOG"} = "no";
}
if (length($ENV{"TPCD_SMPDEGREE"}) <= 0){
    $ENV{"TPCD_SMPDEGREE"} = 1;
}
if (length($ENV{"TPCD_ACTIVATE"}) <= 0){
    $ENV{"TPCD_ACTIVATE"} = "no";
}
if (length($ENV{"TPCD_APPEND_ON"}) <= 0){
    $ENV{"TPCD_APPEND_ON"}="yes"
}
if
(length($ENV{"TPCD_GENERATE_SEED_FILE"
}) <= 0){
$ENV{"TPCD_GENERATE_SEED_FILE"}="no";
}

#setup global variables
$tpcdVersion=
    $ENV{"TPCD_VERSION"};
$buildStage=
    $ENV{"TPCD_BUILD_STAGE"};
$mode=
    $ENV{"TPCD_MODE"};
$delim =
    $ENV{"TPCD_PATH_DELIM"};
$sep =
    $ENV{"COMMAND_SEP"};
$dddlpath=
    $ENV{"TPCD_DDLPATH"};
$extraindex=
    $ENV{"TPCD_EXTRAINDEX"};
$earlyindex=
    $ENV{"TPCD_EARLYINDEX"};
$loadstats=
    $ENV{"TPCD_LOADSTATS"};
$addRI=
    $ENV{"TPCD_ADD_RI"};
$astFile=
    $ENV{"TPCD_AST"};
$genSeed=
    $ENV{"TPCD_GENERATE_SEED_FILE"};
};
$log=
    $ENV{"TPCD_LOG"};

$activate=
    $ENV{"TPCD_ACTIVATE"};
$RealAudit=
    $ENV{"TPCD_AUDIT"};
$auditDir=
    $ENV{"TPCD_AUDIT_DIR"};
$loadsetScript=
    $ENV{"TPCD_LOAD_DB2SET_SCRIPT"};
};
$user=
    $ENV{"USER"};
$logDirScript=
    $ENV{"TPCD_LOG_DIR_SETUP_SCRIP
T"};
$logprimary=
    $ENV{"TPCD_LOGPRIMARY"};
$logsecond=
    $ENV{"TPCD_LOGSECOND"};
$logfilsiz=
    $ENV{"TPCD_LOGFILSIZ"};
$dbpath =
    $ENV{"TPCD_DBPATH"};
$explainDDL=
    $ENV{"TPCD_EXPLAIN_DDL"};
$platform=
    $ENV{"TPCD_PLATFORM"};
$buffpooldef=
    $ENV{"TPCD_BUFFERPOOL_DEF"};
$stagingTbl =
    $ENV{"TPCD_STAGING_TABLE_DDL"}
};
$preloadSampleUF=
    $ENV{"TPCD_PRELOAD_STAGING_T
ABLE_SCRIPT"};
$deleteSampleUF=
    $ENV{"TPCD_DELETE_STAGING_TA
BLE_SQL"};
$machine=
    $ENV{"TPCD_MACHINE"};
$runstatShort =
    $ENV{"TPCD_RUNSTATSHORT"};
$runstats =
    $ENV{"TPCD_RUNSTATS"};
$smpdegree =
    $ENV{"TPCD_SMPDEGREE"};
$agentpri =
    $ENV{"TPCD_AGENTPRI"};
$setScript =
    $ENV{"TPCD_DB2SET_SCRIPT"};
$backupdir =
    $ENV{"TPCD_BACKUP_DIR"};
$nodegroupdef=
    $ENV{"TPCD_NODEGROUP_DEF"};
$dbgen=
    $ENV{"TPCD_DBGEN"};
$appendOn=
    $ENV{"TPCD_APPEND_ON"};

```



```

$indexddl=
    $ENV{"TPCD_INDEXDDL"};

if($qual eq "QUAL"){
    $logDir=
        $ENV{"TPCD_LOG_QUAL_DIR"};
    $dbname=
        $ENV{"TPCD_QUAL_DBNAME"};
    $input=
        $ENV{"TPCD_QUAL_INPUT"};
    $sf=
        $ENV{"TPCD_QUAL_SF"};
    $loadconfigfile=$ENV{"TPCD_LOAD_Q
UALCONFIGFILE"};
    $loadDBMconfig=
        $ENV{"TPCD_LOAD_DBM_QUALCON
FIGFILE"};
    $loadscript =
        $ENV{"TPCD_LOAD_SCRIPT_QUAL"};
    $configfile =
        $ENV{"TPCD_QUALCONFIGFILE"};
    $dbmconfig =
        $ENV{"TPCD_DBM_QUALCONFIG"};
    $ddl=
        $ENV{"TPCD_QUAL_DDL"};
    $bspddl=
        $ENV{"TPCD_QUAL_TBSP_DDL"};
}
else{
    $logDir=
        $ENV{"TPCD_LOG_DIR"};
    $dbname=
        $ENV{"TPCD_DBNAME"};
    $input=
        $ENV{"TPCD_INPUT"};
    $sf=
        $ENV{"TPCD_SF"};
    $loadconfigfile=$ENV{"TPCD_LOAD_C
ONFIGFILE"};
    $loadDBMconfig=
        $ENV{"TPCD_LOAD_DBM_CONFIGFI
LE"};
    $loadscript =
        $ENV{"TPCD_LOAD_SCRIPT"};
    $configfile =
        $ENV{"TPCD_CONFIGFILE"};
    $dbmconfig =
        $ENV{"TPCD_DBM_CONFIG"};
    $ddl=
        $ENV{"TPCD_DDL"};
    $bspddl=
        $ENV{"TPCD_TBSP_DDL"};
}

if (( $mode eq "uni" ) || ( $mode eq "smp" )){
    $all_in="once";
    $all_pn="once";
    $once="once";
}
else{
    $all_in="all_in";
    $all_pn="all_pn";
    $once="once";
}
#-----#
-----#
# echo parameter settings to acknowledge what is
being built #
# and set db2set options for database load
#
#-----#
-----#
&printSummary;
print "\nSleeping for 15 seconds to give you a
chance to reconsider...\n";
sleep 15;
if ( $platform eq "nt" ){
    if (($mode eq "uni") || ($mode eq "smp")){
        #spaces required for NT
        $rc=&doddb_noconn("db2set DB2OPTIONS=\
-t -v +c\";db2set DB2NTNOCACHE=ON",$all_in);
    }
    else{
        $rc=&doddb_noconn("db2set
DB2OPTIONS=\\\ " -t -v +c\\\";db2set
DB2NTNOCACHE=ON",$all_in);
    }
}
else{
    if (($mode eq "uni") || ($mode eq "smp")){
        $rc=&doddb_noconn("db2set DB2OPTIONS=\
-t -v +c\\"", $all_in);
    }
    else{
        $rc=&doddb_noconn("db2set
DB2OPTIONS=\\\ "-t -v +c\\\\"", $all_in);
    }
}
if ( $rc != 0 ){
    die "failure setting db2 environment variable : rc =
$rc\n";
}
#-----#
-----#
# set the db2 env vars for loading, from the
TPCD_LOAD_DB2SET_SCRIPT script #
#-----#
-----#

```

```

if ( $loadsetScript ne "NULL" )
{
  if ( $platform eq "nt" ){
    if ( ( $mode eq "uni" ) || ( $mode eq "smp" ) ){

$src=system("${ddlpath}${delim}$loadsetScript");
    }
    else{
      $src=system(" rah \" cd ${ddlpath} &
loadsetScript\" ");
    }
  }
  else{

$src=system("${ddlpath}${delim}$loadsetScript");
  }
  ($src == 0) || die "failure loading db2set parms
from $loadsetScript \n";
}

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

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

$src = &setLoadConfig;

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

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

```

```

!&createTablespaces || print "WARNING:
create tablespaces error.\n";
!&createExplainTbls || print "ERROR:
create EXPLAIN tables failed.\n";
!&createTables || print "ERROR: create
tables failed.\n";

mkdir("${delim}tmp${delim}$instance",07
77);

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

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

  $filename="$auditDir${delim}$dbname.$u
ser.update.pair.num";
} else{
  mkdir (" $auditDir", 0775) || die
"cannot mkdir $auditDir";
}
print "setting update pair num to 1\n";
system("echo 1 > $filename");
};
#-----#
# Begin build from Index or Load
#
#-----#

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

# if indexes haven't been created, do so
now
if ( $earlyindex ne "yes" ){

```

```

                !&createIndexes("normal") || die
"ERROR: create indexes failed.\n";
            }
            if ( $extraindex ne "no" ){
                !&createIndexes("extra") || die
"ERROR: create extra indexes failed.\n";
            }
}; # end create/load/index phase of the build

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

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

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

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

    &dodb2file($dbname,"$ddlpath${delim}$addRI", $o
nce);
    &outtime("*** Adding RI constraints completed");
}

#add the AST if it has been requested

```

```

if ( $astFile ne "NULL" ){
    &outtime("*** Adding AST started");

    &dodb2file($dbname,"$ddlpath${delim}$astFile", $o
nce);
    &outtime("*** Adding AST completed");
}

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

# set the configuration
&outtime("*** Set Configuration started");
&outtime("*** Setting degree of parallelism");

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

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

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

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

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

```

```

# if we are in the real audit mode then run dbtables
and dbcheck before #
# we print out the notice that we're ready to run
performance tests #
# if we are building the qualification database then
we'll bind to both #
# the dbname database and the qualification
database #
#-----#
-----#

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

if ( $RealAudit eq "yes" ){
    &rm("$auditDir${delim}tools${delim}tpcd
.runsetup");
    system("perl setupRun");
    if ( $qual eq "QUAL" ){
        $verifyType="q";
    }
    else{
        $verifyType="t";
    }
    system("perl tablesdb $verifyType");
    &dodb2file($dbname,"$auditDir${delim}to
ols${delim}first10rows.sql",$once);
}

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

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

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

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

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

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

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

```

```

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

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

#-----#
# Function: createNodegroups
#
#-----#
sub createNodegroups{
    &outtime("*** Creating the nodegroups.");
    my $src;
    if ( $nodegroupdef ne "NULL"){

```

```

        $src =
&dodb2file($dbname, "$ddlpath${delim}$nodegroup
pdef", $once);
    }
}

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

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

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

```

```

        db2 alter table explain_predicate locksize
table append on $sep \
        db2 alter table explain_stream locksize
table append on",
        $once);
}
#-----#
# Function: createBufferPools
#
#-----#
sub createBufferPools{
    my $src;
    &outtime("*** Creating the bufferpools");

    if ( $buffpooldef ne "NULL" ){
        #run the create bufferpool ddl

        $src =
&dodb2file($dbname,"$ddlpath${delim}$buffpooldef",
$once);
    }
}
#-----#
# Function: createTablespaces
#
#-----#
sub createTablespaces{
    &outtime("*** Ready to start creating the
tablespaces");
    # setup required variables
    my $src;
    $src =
&dodb2file($dbname,"$ddlpath${delim}$tbspddl",
$once);
    ($src == 0) || return $src;
    # create/populate the staging tables
    if ( $stagingTbl ne "NULL" ){
        # staging tables must be created
for both test and qualification database
        # but they do not need to be
populated for the qualification database
        $src =
&dodb2file($dbname,"$ddlpath${delim}$stagingTbl",
$once);
        ($src == 0) || return $src;
        if ( $qual ne "QUAL" ){
            if ( $preloadSampleUF ne
"NULL" ){
                # preload the
sample UF data for statistics gathering
                $src = system
("perl $ddlpath${delim}$preloadSampleUF");
                #($src == 0) ||
return $src;
            }
            if ( $deleteSampleUF ne
"NULL" ){
                # delete the
sample rows now that stats have been gathered
                $src =
&dodb2file($dbname,"$ddlpath${delim}$deleteSam
pleUF",$once);
                #($src == 0) ||
return $src;
            }
        }
    }
}
#-----#
# Function: createTables
#
#-----#
sub createTables{
    my $src;
    $src =
&dodb2file($dbname,"$ddlpath${delim}$ddl",
$once);
    ($src == 0) || return $src;
    # update the locksize on the non-updated
tables to be table level locking
    # update the tables for appendmode
    if ($appendOn eq "yes"){
        $src = &dodb_conn($dbname,
"db2 alter table tpcd.nation
locksize table $sep \
        db2 alter table tpcd.region
locksize table $sep \
        db2 alter table tpcd.customer
locksize table $sep \
        db2 alter table tpcd.supplier
locksize table $sep \
        db2 alter table tpcd.part locksize
table $sep \
        db2 alter table tpcd.partsupp
locksize table $sep \
        db2 alter table tpcd.lineitem
append on $sep \
        db2 alter table tpcd.orders append
on",
        $once);
    }
    else{
        $src = &dodb_conn($dbname,

```

```

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

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

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

```

```

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

        if ($loadconfigfile eq "NULL"){
            if ( $machine eq "small" ){
                $buffpage = 5000;
                $sortheap = 3000;
                $sheapthres = 8000;
                $util_heap_sz = 5000;
                $ioservers = 6;
            }
            elsif ( $machine eq "medium" ){
                $buffpage = 10000;
                $sortheap = 8000;
                $sheapthres = 20000;
                $util_heap_sz = 10000;
                $ioservers = 10;
            }
            elsif ( $machine eq "big" ){
                $buffpage = 30000;
                $sortheap = 20000;
                $sheapthres = 50000;
                $util_heap_sz = 30000;
                $ioservers = 20;
            }
            else {
                die "Neither a LOAD
config filename nor a valid machine size has \
been specified!\n";
            }
            $src = &dodb_noconn("db2 update
db cfg for $dbname using buffpage $buffpage $sep \
db2 update db cfg for $dbname
using sortheap $sortheap $sep \
db2 update db cfg for $dbname
using num_iocleaners $ioclnrs $sep \
db2 update db cfg for $dbname
using num_ioservers $ioservers $sep \
db2 update db cfg for $dbname
using util_heap_sz $util_heap_sz $sep \
db2 update db cfg for $dbname
using chngpgs_thresh $chnngpgs",$all_ln);
        }
    }
}
else{

```

```

                $rc =
&dodb2file_noconn("$ddlpath${delim}$loadconfigfile", $all_in);
            }
            ($rc == 0) || return $rc;
            if($loadDBMconfig ne "NULL"){
                $rc =
&dodb2file_noconn("$ddlpath${delim}$loadDBMconfig", $once);
            }
            else{
                $rc = &dodb_noconn("db2 update dbm cfg using sheapthres $sheapthres", $once);
            }
            ($rc == 0) || return $rc;
            &dodb_noconn("db2 terminate", $once);
            $rc = &stopStart;
            return $rc;
        }
#-----#
# Function: loadData
#
#-----#
sub loadData{
    # start the dbgen and load.....call the
    # specific mode for loading (uni,smp,mln)
    my $rc;
    if (( $mode eq "uni" ) || ( $mode eq "smp"
))){
        &outtime("*** Starting the load");
        # call the appropriate dbgen/load
        for uni/smp
            if ( $loadscript eq "NULL"){
                $rc = system("perl
genloaduni $qual");
                ($rc == 0) || print
"ERROR: genloaduni failed rc = $rc\n";
            }
            else{
                $rc =
&dodb2file_noconn("$ddlpath${delim}$loadscript",
$once);
                ($rc == 0) || print
"ERROR: load script: $loadscript 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
                if ( $loadscript eq "NULL"){
                    $rc = system("perl
genloadmpp $qual");
                    ($rc == 0) || print
"ERROR: genloadmpp failed. rc = $rc\n";
                }
                else{
                    system("$ddlpath${delim}$loadscript");
                    # $rc =
&dodb2file_noconn("$ddlpath${delim}$loadscript
$sf");
                    # ($rc == 0) || print
"ERROR: load script $loadscript failed. rc = $rc\n";
                }
                else{
                    print "TPCD_MODE not set to
one of uni, smp, mln or mpp\n";
                    $rc = 1;
                }
                ($rc == 0) || &outtime("*** Load
complete");
                return $rc;
            }
}
#-----#
# Function: doRunStats
#
#-----#
sub doRunStats{
    # if loadstats not gathered, then index stats not
    # gathered either.
    &outtime("*** Runstats started");
    if ( $runstatShort ne "NULL" ){
        # we've specified a second runstats file...This
        # runstats file should do
        # runstats for all table except lineitem. The
        # lineitem runstats command
        # should be left in the main runstats file.
        if ( $platform eq "aix" || $platform eq "sun" ||
$platform eq "ptx" ){
            print "runstats from
$ddlpath${delim}$runstatShort running now\n";
            $rc = system("db2 -tvf
\"$ddlpath${delim}$runstatShort\" >
\"$auditDir${delim}tools${delim}runstatShort.out\"
& ");
            print "rc from runstatshort=$rc\n";
        }
        elsif ( $platform eq "nt" ){
            system("start db2 -tvf
$ddlpath${delim}$runstatShort");
        }
        else
        {

```



```

        print "Don't know how to start in
background on $platform platform\n";
        print "therefore running runstats
serially\n";

&dodb2file($dbname,"$ddlpath${delim}$runstatSh
ort", $once);
    }
}
# run the full runstats, or the remainder of what
wasn't put into the short
# runstats file. You should be sure that this
runstats will take longer
# than the short runstats that is running in the
background, otherwise
# setting the config will happen before this is
done.

&dodb2file($dbname,"$ddlpath${delim}$runstats",
$once);
    &outtime("*** Runstats completed");
}

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

    if ( $agentpri ne "NULL" ){
        &dodb_noconn("db2 update dbm cfg
using AGENTPRI $agentpri", $once);
    }
    # set the db2 environment variables for
running the benchmark
    if ( $setScript ne "NULL" ){
        if ( $platform eq "aix" || $platform eq
"sun" || $platform eq "ptx" ){
            $ret=system("${ddlpath}${delim}$setScript");
        }
        elsif ( $platform eq "nt" ){

```

```

        if (($mode eq "uni" ) || ($mode eq
"smp" )){
            $ret = system("perl
${ddlpath}${delim}$setScript");
        }
        else{
            $ret = system(" rah \" cd ${ddlpath}
& $setScript\" ");
        }
    }
    #($ret == 0 ) || die "failure setting
runtime db2set parms from $setScript \n";
}

#-----#
# Function: createBackup
#
#-----#
sub createBackup{
    my $src;
    &dodb_noconn("db2 update database
configuration for $dbname using LOGRETAIN
yes", $all_in);
    print "\n NOTE: DO NOT RESET THE
DATABASE CONFIGURATION or you will lose
logretain\n";
    # force a connection to the database on all
nodes to ensure LOGRETAIN is
    # set in effect.
    # An error message will print to screen if
the logretain is set properly
    # i.e. SQL116N A connection to or
activation of database <database name>
    # cannot be made.
    # This is expected and the lack of this error
message should be seen as an
    # error in the database build.
    &dodb_conn($dbname,"db2 \"select
count(*) from tpcd.region\"", $all_in);

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

```

```

                Src=system("db2_all \|\|}<<-
000< db2 backup database $dbname to $backupdir
without prompting\ ');
                (Src == 0) || print "ERROR:
backup of remaining nodes failed rc = $rc\n";
                }
                else{
                Src = &dodb_noconn("db2
backup database $dbname to $backupdir,$once);
                }
                (Src == 0) || &outtime("***
Finished the backup");
                }
                else{
                # This is the test database. Clause
3.1.4 states that "the test sponsor is
                # not required to make or have
backup copies of the test database; however
                # all other mechanisms that
guarantee durability of the qualification
                # database must be enabled in the
same way for the test database".
                # According to this clause we do
need to keep the backup of the database.
                Src = &dodb_noconn("db2dart
$dbname /CHST /WHAT DBBP OFF",$all_in);
                }
                return $rc;
        }

#-----#
# Function: printSummary
#
#-----#
sub printSummary{
        if ( $buildStage ne "ALL" ){
                print "***** STARTING the build
process at the $buildStage Stage *****\n";
        }
        print "Building a TPC-D Version
$tpcdVersion $sf GB database on $dbpath with: \n";
        print " Mode = $mode \n";
        print " Tablespace ddl in
$ddlpath${delim}$tbspddl \n";
        if ( $nodegroupdef ne "NULL" ){
                print " Nodegroup ddl in
$ddlpath${delim}$nodegroupdef \n";
        }
        if ( $buffpooldef ne "NULL" ){
                print " Bufferpool ddl in
$ddlpath${delim}$buffpooldef \n";
        }
        print " Table ddl in $ddlpath${delim}$ddl
\n";

```

```

        print " Index ddl in
$ddlpath${delim}$indexddl\n";
        if ( $extraindex ne "no" ){
                print " Indices to create after the load
$ddlpath${delim}$extraindex\n";
        }
        if ( $loadscript eq "NULL"){
                if ( $input eq "NULL" ){
                        print " Data generated by DBGEN in
$dbngen\n";
                }
                else{
                        print " Data loaded from flat files in
$input\n";
                }
        }
        if ( $earlyindex eq "yes" ){
                print " Indexes created before loading\n";
        }
        else{
                print " Indexes created after loading\n";
        }
        if ( $addRI ne "NULL" ){
                print " RI being used from
$ddlpath${delim}$addRI\n";
        }
        if ( $astFile ne "NULL" ){
                print " AST being used from
$ddlpath${delim}$astFile\n";
        }
        if ( $loadstats eq "yes" ){
                if ( $earlyindex eq "yes" ){
                        print " Statistics for tables and indexes
gathered during load\n";
                }
                else{
                        if ( $runstatShort eq "NULL" ){
                                print " Statistics for tables and indexes
gathered after load using $ddlpath${delim}$runstats
\n";
                        }
                        else{
                                print " Statistics for tables and
indexes gathered after load using
$ddlpath${delim}$runstats and
$ddlpath${delim}$runstatShort\n";
                        }
                }
        }
        else{
                if ( $runstatShort eq "NULL" ){
                        print " Statistics for tables and indexes
gathered after load using $ddlpath${delim}$runstats
\n";
                }
                else{

```

```

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

```

```

        print " Log Primary left at default\n";
    }
else{
    print " Log Primary set to
$logprimary\n";
}
if ( $logsecond eq "NULL" ){
    print " Log Second left at default\n";
}
else{
    print " Log second set to $logsecond\n";
}
if ( $logfilsiz eq "NULL" ){
    print " Logfilsiz left at default\n";
}
else{
    print " Logfilsiz set to $logfilsiz\n";
}
if (($loadconfigfile eq "") || ($loadconfigfile
eq "NULL")){
    print " Machine size set to $machine so
the following 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 =
$chnpggs\n";
    print " UTIL_HEAP_SZ =
$util_heap_sz\n";
    print " Degree of parallelism
(dft_degree and max_querydegree) set to
$smpdegree\n";
    print " Parameters for load are: temp
file = $ldtemp\n";
    print " sort buf =
$sortbuf\n";
    print " ld parallelism =
$load_parallelism\n";
    if ( $fparse eq "yes" ){
        print " FASTPARSE
used on load\n";
    }
}
if ( $loadscript ne "NULL" ){
    print " Load commands in
$ddlpath${delim}$loadscript\n";
}
print " Degree of parallelism (dft_degree
and max_querydegree) set to $smpdegree\n";

```

```

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

1;

create_bufferpools

-----
-- Create Bufferpools
-----
ALTER BUFFERPOOL IBMDEFAULTBP SIZE
200;
COMMIT WORK;
CREATE BUFFERPOOL BP32K ALL NODES
SIZE 40000 PAGESIZE 32K;
COMMIT WORK;
ALTER BUFFERPOOL BP32K
NUMBLOCKPAGES 8000 BLOCKSIZE 16;
COMMIT WORK;
CREATE BUFFERPOOL BP8K ALL NODES
SIZE 36000 PAGESIZE 8K;
COMMIT WORK;
ALTER BUFFERPOOL BP8K
NUMBLOCKPAGES 7200 BLOCKSIZE 64;
COMMIT WORK;

create_indexes

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

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

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

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

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

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

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

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

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

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

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

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

values(current timestamp);

```

```

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

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

values(current timestamp);

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

create_nodegroups

-----
-- Create Nodegroups
-----
CREATE NODEGROUP ng_all ON NODES (0 to
15);
CREATE NODEGROUP ng_node0 ON NODE (0);

COMMIT WORK;

create_tables

-----
-- Create Tables
-----

CREATE TABLE TPCD.NATION (
N_NATIONKEY INTEGER NOT NULL,

                N_NAME      CHAR(25) NOT
NULL,

                N_REGIONKEY INTEGER NOT
NULL,

                N_COMMENT
VARCHAR(152) NOT NULL)

                IN SMALL_DATA;

```

```

CREATE TABLE TPCD.REGION (
R_REGIONKEY INTEGER NOT NULL,

                R_NAME      CHAR(25) NOT
NULL,

                R_COMMENT
VARCHAR(152) NOT NULL)

                IN SMALL_DATA;

CREATE TABLE TPCD.PART ( P_PARTKEY
INTEGER NOT NULL,

                P_NAME      VARCHAR(55)
NOT NULL,

                P_MFGR      CHAR(25) NOT
NULL,

                P_BRAND     CHAR(10) NOT
NULL,

                P_TYPE      VARCHAR(25) NOT
NULL,

                P_SIZE      INTEGER NOT
NULL,

                P_CONTAINER CHAR(10) NOT
NULL,

                P_RETAILPRICE FLOAT NOT
NULL,

                P_COMMENT   VARCHAR(23)
NOT NULL )

                IN DATA_INDEX

                PARTITIONING KEY(P_PARTKEY)
USING HASHING;

CREATE TABLE TPCD.SUPPLIER (
S_SUPPKEY INTEGER NOT NULL,

                S_NAME      CHAR(25) NOT
NULL,

```

NOT NULL,	S_ADDRESS VARCHAR(40)	NULL,	C_PHONE CHAR(15) NOT
NOT NULL,	S_NATIONKEY INTEGER	NULL,	C_ACCTBAL FLOAT NOT
NULL,	S_PHONE CHAR(15) NOT	NOT NULL,	C_MKTSEGMENT CHAR(10)
NULL,	S_ACCTBAL FLOAT NOT		C_COMMENT
	S_COMMENT		VARCHAR(117) NOT NULL)
	VARCHAR(101) NOT NULL)		IN DATA_INDEX
	IN DATA_INDEX		PARTITIONING KEY(C_CUSTKEY) USING
	PARTITIONING KEY(S_SUPPKEY) USING		HASHING;
	HASHING;		CREATE TABLE TPCD.ORDERS (
CREATE TABLE TPCD.PARTSUPP (PS_PARTKEY INTEGER NOT NULL,		O_ORDERKEY INTEGER NOT NULL,
NULL,	PS_SUPPKEY INTEGER NOT		O_CUSTKEY INTEGER NOT
NOT NULL,	PS_AVAILQTY INTEGER		NULL,
NOT NULL,	PS_SUPPLYCOST FLOAT		O_ORDERSTATUS CHAR(1)
	PS_COMMENT		O_TOTALPRICE FLOAT NOT
	VARCHAR(199) NOT NULL)		NULL,
	IN DATA_INDEX		O_ORDERDATE DATE NOT
	PARTITIONING KEY(PS_PARTKEY)		NULL,
	USING HASHING;		O_ORDERPRIORITY CHAR(15)
CREATE TABLE TPCD.CUSTOMER (C_CUSTKEY INTEGER NOT NULL,		NOT NULL,
NOT NULL,	C_NAME VARCHAR(25)		O_CLERK CHAR(15) NOT
NOT NULL,	C_ADDRESS VARCHAR(40)		NULL,
NOT NULL,	C_NATIONKEY INTEGER		O_SHIPPRIORITY INTEGER
			NOT NULL,
			O_COMMENT
			VARCHAR(79) NOT NULL)
			ORGANIZE BY (O_ORDERDATE)
			IN DATA_INDEX
			PARTITIONING KEY(O_ORDERKEY)
			USING HASHING;

```

CREATE TABLE TPCD.LINEITEM (
L_ORDERKEY  INTEGER NOT NULL,

                L_PARTKEY  INTEGER NOT
NULL,

                L_SUPPKEY  INTEGER NOT
NULL,

                L_LINENUMBER  INTEGER
NOT NULL,

                L_QUANTITY  FLOAT NOT
NULL,

                L_EXTENDEDPRICE  FLOAT
NOT NULL,

                L_DISCOUNT  FLOAT NOT
NULL,

                L_TAX      FLOAT NOT
NULL,

                L_RETURNFLAG  CHAR(1)
NOT NULL,

                L_LINESTATUS  CHAR(1) NOT
NULL,

                L_SHIPDATE  DATE NOT
NULL,

                L_COMMITDATE  DATE NOT
NULL,

                L_RECEIPTDATE  DATE NOT
NULL,

                L_SHIPINSTRUCT  CHAR(25)
NOT NULL,

                L_SHIPMODE   CHAR(10)
NOT NULL,

                L_COMMENT
VARCHAR(44) NOT NULL)
ORGANIZE BY (L_SHIPDATE)

        IN DATA_INDEX

        PARTITIONING KEY(L_ORDERKEY)
USING HASHING;

```

```
COMMIT WORK;
```

```
create_tablespace
```

```
-----
-- Create Tablespaces
-----
```

```

CREATE temporary TABLESPACE tempspace
PAGE SIZE 8K
MANAGED BY database
USING (
    DEVICE '/dev/raw/raw21' 3000M,
    DEVICE '/dev/raw/raw22' 3000M,
    DEVICE '/dev/raw/raw23' 3000M,
    DEVICE '/dev/raw/raw24' 3000M,
    DEVICE '/dev/raw/raw25' 3000M,
    DEVICE '/dev/raw/raw26' 3000M,
    DEVICE '/dev/raw/raw27' 3000M,
    DEVICE '/dev/raw/raw28' 3000M,
    DEVICE '/dev/raw/raw29' 3000M,
    DEVICE '/dev/raw/raw30' 3000M,
    DEVICE '/dev/raw/raw31' 3000M,
    DEVICE '/dev/raw/raw32' 3000M,
    DEVICE '/dev/raw/raw33' 3000M,
    DEVICE '/dev/raw/raw34' 3000M,
    DEVICE '/dev/raw/raw35' 3000M,
    DEVICE '/dev/raw/raw36' 3000M,
    DEVICE '/dev/raw/raw37' 3000M,
    DEVICE '/dev/raw/raw38' 3000M,
    DEVICE '/dev/raw/raw39' 3000M,
    DEVICE '/dev/raw/raw40' 3000M
) ON NODE (0)
USING (
    DEVICE '/dev/raw/raw41' 3000M,
    DEVICE '/dev/raw/raw42' 3000M,
    DEVICE '/dev/raw/raw43' 3000M,
    DEVICE '/dev/raw/raw44' 3000M,
    DEVICE '/dev/raw/raw45' 3000M,
    DEVICE '/dev/raw/raw46' 3000M,
    DEVICE '/dev/raw/raw47' 3000M,
    DEVICE '/dev/raw/raw48' 3000M,
    DEVICE '/dev/raw/raw49' 3000M,
    DEVICE '/dev/raw/raw50' 3000M,
    DEVICE '/dev/raw/raw51' 3000M,
    DEVICE '/dev/raw/raw52' 3000M,
    DEVICE '/dev/raw/raw53' 3000M,
    DEVICE '/dev/raw/raw54' 3000M,
    DEVICE '/dev/raw/raw55' 3000M,
    DEVICE '/dev/raw/raw56' 3000M,
    DEVICE '/dev/raw/raw57' 3000M,
    DEVICE '/dev/raw/raw58' 3000M,
    DEVICE '/dev/raw/raw59' 3000M,
    DEVICE '/dev/raw/raw60' 3000M
)

```



```

DEVICE '/dev/raw/raw23' 3000M,
DEVICE '/dev/raw/raw24' 3000M,
DEVICE '/dev/raw/raw25' 3000M,
DEVICE '/dev/raw/raw26' 3000M,
DEVICE '/dev/raw/raw27' 3000M,
DEVICE '/dev/raw/raw28' 3000M,
DEVICE '/dev/raw/raw29' 3000M,
DEVICE '/dev/raw/raw30' 3000M,
DEVICE '/dev/raw/raw31' 3000M,
DEVICE '/dev/raw/raw32' 3000M,
DEVICE '/dev/raw/raw33' 3000M,
DEVICE '/dev/raw/raw34' 3000M,
DEVICE '/dev/raw/raw35' 3000M,
DEVICE '/dev/raw/raw36' 3000M,
DEVICE '/dev/raw/raw37' 3000M,
DEVICE '/dev/raw/raw38' 3000M,
DEVICE '/dev/raw/raw39' 3000M,
DEVICE '/dev/raw/raw40' 3000M
) ON NODE (12)
USING (
  DEVICE '/dev/raw/raw41' 3000M,
  DEVICE '/dev/raw/raw42' 3000M,
  DEVICE '/dev/raw/raw43' 3000M,
  DEVICE '/dev/raw/raw44' 3000M,
  DEVICE '/dev/raw/raw45' 3000M,
  DEVICE '/dev/raw/raw46' 3000M,
  DEVICE '/dev/raw/raw47' 3000M,
  DEVICE '/dev/raw/raw48' 3000M,
  DEVICE '/dev/raw/raw49' 3000M,
  DEVICE '/dev/raw/raw50' 3000M,
  DEVICE '/dev/raw/raw51' 3000M,
  DEVICE '/dev/raw/raw52' 3000M,
  DEVICE '/dev/raw/raw53' 3000M,
  DEVICE '/dev/raw/raw54' 3000M,
  DEVICE '/dev/raw/raw55' 3000M,
  DEVICE '/dev/raw/raw56' 3000M,
  DEVICE '/dev/raw/raw57' 3000M,
  DEVICE '/dev/raw/raw58' 3000M,
  DEVICE '/dev/raw/raw59' 3000M,
  DEVICE '/dev/raw/raw60' 3000M
) ON NODE (13)
USING (
  DEVICE '/dev/raw/raw21' 3000M,
  DEVICE '/dev/raw/raw22' 3000M,
  DEVICE '/dev/raw/raw23' 3000M,
  DEVICE '/dev/raw/raw24' 3000M,
  DEVICE '/dev/raw/raw25' 3000M,
  DEVICE '/dev/raw/raw26' 3000M,
  DEVICE '/dev/raw/raw27' 3000M,
  DEVICE '/dev/raw/raw28' 3000M,
  DEVICE '/dev/raw/raw29' 3000M,
  DEVICE '/dev/raw/raw30' 3000M,
  DEVICE '/dev/raw/raw31' 3000M,
  DEVICE '/dev/raw/raw32' 3000M,
  DEVICE '/dev/raw/raw33' 3000M,
  DEVICE '/dev/raw/raw34' 3000M,

```

```

DEVICE '/dev/raw/raw35' 3000M,
DEVICE '/dev/raw/raw36' 3000M,
DEVICE '/dev/raw/raw37' 3000M,
DEVICE '/dev/raw/raw38' 3000M,
DEVICE '/dev/raw/raw39' 3000M,
DEVICE '/dev/raw/raw40' 3000M
) ON NODE (14)
USING (
  DEVICE '/dev/raw/raw41' 3000M,
  DEVICE '/dev/raw/raw42' 3000M,
  DEVICE '/dev/raw/raw43' 3000M,
  DEVICE '/dev/raw/raw44' 3000M,
  DEVICE '/dev/raw/raw45' 3000M,
  DEVICE '/dev/raw/raw46' 3000M,
  DEVICE '/dev/raw/raw47' 3000M,
  DEVICE '/dev/raw/raw48' 3000M,
  DEVICE '/dev/raw/raw49' 3000M,
  DEVICE '/dev/raw/raw50' 3000M,
  DEVICE '/dev/raw/raw51' 3000M,
  DEVICE '/dev/raw/raw52' 3000M,
  DEVICE '/dev/raw/raw53' 3000M,
  DEVICE '/dev/raw/raw54' 3000M,
  DEVICE '/dev/raw/raw55' 3000M,
  DEVICE '/dev/raw/raw56' 3000M,
  DEVICE '/dev/raw/raw57' 3000M,
  DEVICE '/dev/raw/raw58' 3000M,
  DEVICE '/dev/raw/raw59' 3000M,
  DEVICE '/dev/raw/raw60' 3000M
) ON NODE (15)
BUFFERPOOL BP8K
EXTENTSIZ 64
PREFETCHSIZE 384
OVERHEAD 5.7
TRANSFERRATE .61;
COMMIT WORK;

CREATE regular TABLESPACE small_data
  IN NODEGROUP ng_node0
  PAGESIZE 4K
  MANAGED BY system
USING (
  'data/small_data'
)
BUFFERPOOL ibmdefaultbp
OVERHEAD 25
TRANSFERRATE .3;
COMMIT WORK;

CREATE regular TABLESPACE data_index
  IN NODEGROUP ng_all
  PAGESIZE 32K
  MANAGED BY database
USING (
  DEVICE '/dev/raw/raw1' 3000M,
  DEVICE '/dev/raw/raw2' 3000M,
  DEVICE '/dev/raw/raw3' 3000M,

```



```

DEVICE '/dev/raw/raw18' 3000M,
DEVICE '/dev/raw/raw19' 3000M,
DEVICE '/dev/raw/raw20' 3000M
) ON NODE (9)
USING (
  DEVICE '/dev/raw/raw1' 3000M,
  DEVICE '/dev/raw/raw2' 3000M,
  DEVICE '/dev/raw/raw3' 3000M,
  DEVICE '/dev/raw/raw4' 3000M,
  DEVICE '/dev/raw/raw5' 3000M,
  DEVICE '/dev/raw/raw6' 3000M,
  DEVICE '/dev/raw/raw7' 3000M,
  DEVICE '/dev/raw/raw8' 3000M,
  DEVICE '/dev/raw/raw9' 3000M,
  DEVICE '/dev/raw/raw10' 3000M
) ON NODE (10)
USING (
  DEVICE '/dev/raw/raw11' 3000M,
  DEVICE '/dev/raw/raw12' 3000M,
  DEVICE '/dev/raw/raw13' 3000M,
  DEVICE '/dev/raw/raw14' 3000M,
  DEVICE '/dev/raw/raw15' 3000M,
  DEVICE '/dev/raw/raw16' 3000M,
  DEVICE '/dev/raw/raw17' 3000M,
  DEVICE '/dev/raw/raw18' 3000M,
  DEVICE '/dev/raw/raw19' 3000M,
  DEVICE '/dev/raw/raw20' 3000M
) ON NODE (11)
USING (
  DEVICE '/dev/raw/raw1' 3000M,
  DEVICE '/dev/raw/raw2' 3000M,
  DEVICE '/dev/raw/raw3' 3000M,
  DEVICE '/dev/raw/raw4' 3000M,
  DEVICE '/dev/raw/raw5' 3000M,
  DEVICE '/dev/raw/raw6' 3000M,
  DEVICE '/dev/raw/raw7' 3000M,
  DEVICE '/dev/raw/raw8' 3000M,
  DEVICE '/dev/raw/raw9' 3000M,
  DEVICE '/dev/raw/raw10' 3000M
) ON NODE (12)
USING (
  DEVICE '/dev/raw/raw11' 3000M,
  DEVICE '/dev/raw/raw12' 3000M,
  DEVICE '/dev/raw/raw13' 3000M,
  DEVICE '/dev/raw/raw14' 3000M,
  DEVICE '/dev/raw/raw15' 3000M,
  DEVICE '/dev/raw/raw16' 3000M,
  DEVICE '/dev/raw/raw17' 3000M,
  DEVICE '/dev/raw/raw18' 3000M,
  DEVICE '/dev/raw/raw19' 3000M,
  DEVICE '/dev/raw/raw20' 3000M
) ON NODE (13)
USING (
  DEVICE '/dev/raw/raw1' 3000M,
  DEVICE '/dev/raw/raw2' 3000M,
  DEVICE '/dev/raw/raw3' 3000M,

```

```

DEVICE '/dev/raw/raw4' 3000M,
DEVICE '/dev/raw/raw5' 3000M,
DEVICE '/dev/raw/raw6' 3000M,
DEVICE '/dev/raw/raw7' 3000M,
DEVICE '/dev/raw/raw8' 3000M,
DEVICE '/dev/raw/raw9' 3000M,
DEVICE '/dev/raw/raw10' 3000M
) ON NODE (14)
USING (
  DEVICE '/dev/raw/raw11' 3000M,
  DEVICE '/dev/raw/raw12' 3000M,
  DEVICE '/dev/raw/raw13' 3000M,
  DEVICE '/dev/raw/raw14' 3000M,
  DEVICE '/dev/raw/raw15' 3000M,
  DEVICE '/dev/raw/raw16' 3000M,
  DEVICE '/dev/raw/raw17' 3000M,
  DEVICE '/dev/raw/raw18' 3000M,
  DEVICE '/dev/raw/raw19' 3000M,
  DEVICE '/dev/raw/raw20' 3000M
) ON NODE (15)
BUFFERPOOL BP32K
EXTENTSIZE 16
PREFETCHSIZE 160
OVERHEAD 25
TRANSFERRATE .3;
COMMIT WORK;

```

createuftbls

```

-----
-- Create Update Function Tables
-----
CREATE TABLE TPCDTEMP.ORDERS_NEW (
  APP_ID INTEGER NOT NULL,
          O_ORDERKEY    INTEGER
NOT NULL,
          O_CUSTKEY     INTEGER NOT
NULL,
          O_ORDERSTATUS CHAR(1)
NOT NULL,
          O_TOTALPRICE  FLOAT NOT
NULL,
          O_ORDERDATE   DATE NOT
NULL,
          O_ORDERPRIORITY CHAR(15)
NOT NULL,
          O_CLERK       CHAR(15) NOT
NULL,
          O_SHIPPRIORITY INTEGER
NOT NULL,
          O_COMMENT
VARCHAR(79) NOT NULL WITH DEFAULT)
PARTITIONING KEY (O_ORDERKEY)
IN DATA_INDEX;

```

```

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

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

CREATE INDEX TPCDTEMP.I_ORDERS_NEW
ON TPCDTEMP.ORDERS_NEW
( APP_ID,
O_ORDERKEY,
O_CUSTKEY,
O_ORDERSTATUS,
O_TOTALPRICE,
O_ORDERDATE,
O_ORDERPRIORITY,
O_CLERK,
O_SHIPPRIORITY,
O_COMMENT);

```

```

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

```

```

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

```

```
COMMIT WORK;
```

db2nodes.cfg

```

0 maguro01 0
1 maguro01 1
2 maguro02 0
3 maguro02 1
4 maguro03 0
5 maguro03 1
6 maguro04 0
7 maguro04 1
8 maguro05 0
9 maguro05 1
10 maguro06 0
11 maguro06 1
12 maguro07 0
13 maguro07 1
14 maguro08 0
15 maguro08 1

```

load_db2set.ksh

```

#!/bin/ksh
db2set DB2OPTIONS="-t -v +c"
db2set DB2_EXTENDED_OPTIMIZATION=Y
db2set DB2_ANTIJOIN=Y
db2set DB2BQTRY=120
db2set DB2RQTIME=30
db2set DB2BPVARS=/home/tpch/custom/bpvar.cfg
db2set DB2_FORCE_FCM_BP=ON
db2set DB2_PARALLEL_IO="*"

```

run_db2set.ksh

```

#!/bin/ksh
db2set DB2OPTIONS="-t -v +c"
db2set DB2_EXTENDED_OPTIMIZATION=Y
db2set DB2_ANTIJOIN=Y
db2set DB2BQTRY=120
db2set DB2RQTIME=30
db2set DB2BPVARS=/home/tpch/custom/bpvar.cfg
db2set DB2_FORCE_FCM_BP=ON
db2set DB2_PARALLEL_IO="*"

```

runstats.ddl

```
RUNSTATS ON TABLE TPCD.NATION WITH
DISTRIBUTION on all columns
and columns (
  n_name like statistics,
  n_comment like statistics )
AND detailed INDEXES ALL;
commit;
RUNSTATS ON TABLE TPCD.REGION WITH
DISTRIBUTION on all columns
and columns (
  r_name like statistics,
  r_comment like statistics )
AND detailed INDEXES ALL;
commit;
RUNSTATS ON TABLE TPCD.SUPPLIER WITH
DISTRIBUTION on all columns
and columns (
  s_name like statistics,
  s_address like statistics,
  s_phone like statistics,
  s_comment like statistics)
AND detailed INDEXES ALL;
commit;
RUNSTATS ON TABLE TPCD.PART WITH
DISTRIBUTION on all columns
and columns (
  p_name like statistics,
  p_mfgr like statistics,
  p_brand like statistics,
  p_type like statistics,
  p_container like statistics,
  p_comment like statistics)
AND detailed INDEXES ALL;
commit;
RUNSTATS ON TABLE TPCD.PARTSUPP
WITH DISTRIBUTION on all columns
and columns (
  ps_comment like statistics)
AND detailed INDEXES ALL;
commit;
RUNSTATS ON TABLE TPCD.CUSTOMER
WITH DISTRIBUTION on all columns
and columns (
  c_name like statistics,
  c_address like statistics,
  c_phone like statistics,
  c_mktsegment like statistics,
  c_comment like statistics)
AND detailed INDEXES ALL;
commit;
RUNSTATS ON TABLE TPCD.ORDERS WITH
DISTRIBUTION on all columns
```

```
and columns (
  o_orderstatus like statistics,
  o_orderpriority like statistics,
  o_clerk like statistics,
  o_comment like statistics)
AND detailed INDEXES ALL;
commit;
RUNSTATS ON TABLE TPCD.LINEITEM WITH
DISTRIBUTION on all columns
and columns (
  l_returnflag like statistics,
  l_linestatus like statistics,
  l_shipinstruct like statistics,
  l_shipmode like statistics,
  l_comment like statistics)
AND detailed INDEXES ALL;
COMMIT WORK;
load_tables.ksh
#!/bin/ksh
messages=${TPCD_TMP_DIR}
rawdata=${TPCD_INPUT}
custom=${TPCD_DDL_PATH}

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

# Nation and Region are loaded into the current
node

db2 connect to tpcd;

echo "Loading Nation at "`date` >>
${messages}/loadstatus.out
db2 "load from ${rawdata}/nation.tbl of del
modified by coldel| fastparse nohead
er messages ${messages}/nation.msg replace into
TPCD.NATION nonrecoverable"

echo "Loading Region at "`date` >>
${messages}/loadstatus.out
db2 "load from ${rawdata}/region.tbl of del
modified by coldel| fastparse nohead
er messages ${messages}/region.msg replace into
TPCD.REGION nonrecoverable"

echo "Loading Partsupp at "`date` >>
${messages}/loadstatus.out
db2 -tvf ${custom}/load_tb_partsupp.ddl
#
echo "Loading Orders at "`date` >>
${messages}/loadstatus.out
db2 -tvf ${custom}/load_tb_orders.ddl
#
```

```

echo "Loading Lineitem at "`date` >>
${messages}/loadstatus.out
db2 -tvf ${custom}/load_tb_lineitem.ddl
#
echo "Loading Customer at "`date` >>
${messages}/loadstatus.out
db2 -tvf ${custom}/load_tb_customer.ddl
#
echo "Loading Part at "`date` >>
${messages}/loadstatus.out
db2 -tvf ${custom}/load_tb_part.ddl
#
echo "Loading Supplier at "`date` >>
${messages}/loadstatus.out
db2 -tvf ${custom}/load_tb_supplier.ddl
#

db2 commit;
db2 terminate;

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

#echo "Starting Sanity Chequing at "`date` >>
${messages}/loadstatus.out
db2 terminate;

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

#echo "Starting Sanity Chequing at "`date` >>
${messages}/loadstatus.out
#db2 connect to tpcd;
#db2 "select count_big(*) as lineitem from
tpcd.lineitem" >> ${messages}/loadsta
tus.out
#db2 "select count_big(*) as orders from
tpcd.orders" >> ${messages}/loadstatus.
out
#db2 "select count_big(*) as partsupp from
tpcd.partsupp" >> ${messages}/loadsta
tus.out
#db2 "select count_big(*) as customer from
tpcd.customer" >> ${messages}/loadsta
tus.out
#db2 "select count_big(*) as part from tpcd.part" >>
${messages}/loadstatus.out
#db2 "select count_big(*) as supplier from
tpcd.supplier" >> ${messages}/loadsta
tus.out
#db2 "select count(*) as nation from tpcd.nation" >>
${messages}/loadstatus.out

```

```

#db2 "select count(*) as region from tpcd.region" >>
${messages}/loadstatus.out
#db2 terminate;
#echo "Finish Sanity Chequing at "`date` >>
${messages}/loadstatus.out

```

load_tb_customer.ddl

```

load from
    customer.tbl
of del
modified by coldel|
    fastparse
    messages /home/tpch/tmp/customer.msg
replace into TPCD.CUSTOMER
nonrecoverable
partitioned db config mode load_only
part_file_location /data
;

```

load_tb_lineitem.ddl

```

load from
    lineitem.tbl.1, lineitem.tbl.2, lineitem.tbl.3
of del
modified by coldel|
    fastparse
    messages /home/tpch/tmp/lineitem.msg
replace into TPCD.LINEITEM
nonrecoverable
partitioned db config mode load_only
part_file_location /data
;

```

load_tb_orders.ddl

```

load from
    orders.tbl
of del
modified by coldel|
    fastparse
    messages /home/tpch/tmp/orders.msg
replace into TPCD.ORDERS
nonrecoverable
partitioned db config mode load_only
part_file_location /data
;

```

load_tb_part.ddl

```

load from
    part.tbl
of del

```

```

modified by coldel|
    fastparse
    messages /home/tpch/tmp/part.msg
replace into TPCD.PART
nonrecoverable
partitioned db config mode load_only
part_file_location /data
;

```

load_tb_partsupp.ddl

```

load from
    partsupp.tbl
of del
modified by coldel|
    fastparse
    messages /home/tpch/tmp/partsupp.msg
replace into TPCD.PARTSUPP
nonrecoverable
partitioned db config mode load_only
part_file_location /data
;

```

load_tb_supplier.ddl

```

load from /data/supplier.tbl
of del
modified by coldel|
    fastparse
    messages /home/tpch/tmp/supplier.msg
replace into TPCD.SUPPLIER
nonrecoverable
CPU_PARALLELISM 4
;

```

load_dbcfg.ddl

```

UPDATE DB CFG FOR TPCD USING DBHEAP
15000 SORTHEAP 25000 SHEAPTHRES_SHR 0
APPGROUP_MEM_SZ 2000 DFT_QUERYOPT 7
DFT_DEGREE 4 NUM_FREQVALUES 0
NUM_QUANTILES 600 LOCKLIST 16384
MAXLOCKS 60 CHNGPGS_THRESH 15
NUM_IOCLEANERS 20 NUM_IOSERVERS 20
MAXFILOP 1024 NEWLOGPATH /home/logs
LOGFILSIZ 10000 LOGPRIMARY 10
LOGSECOND 10 SOFTMAX 750
DATABASE_MEMORY AUTOMATIC
UTIL_HEAP_SZ 50000
;

```

load_dbmcfg.ddl

```

UPDATE DBM CFG USING

```

```

CPUSPEED -1
SHEAPTHRES 150000
MAX_QUERYDEGREE ANY
INTRA_PARALLEL NO
SVCENAME xtpch
NUMDB 1
MAX_TIME_DIFF 1440
DFT_MON_TIMESTAMP OFF
DIAGLEVEL 0
NOTIFYLEVEL 0
;

```

run_dbcfg.ddl

```

UPDATE DB CFG FOR TPCD USING DBHEAP
10000 SORTHEAP 20000 SHEAPTHRES_SHR
250 DATABASE_MEMORY automatic
UTIL_HEAP_SZ 5000 DFT_QUERYOPT 7
DFT_DEGREE 1 NUM_FREQVALUES 0
NUM_QUANTILES 300 LOCKLIST 40000
MAXLOCKS 20 CHNGPGS_THRESH 60
NUM_IOCLEANERS 4 NUM_IOSERVERS 4
MAXFILOP 1024 LOGFILSIZ 50000
LOGPRIMARY 4 LOGSECOND 1 SOFTMAX 360
LOGBUFSZ 2048 MINCOMMIT 2 APPLHEAPSZ
1024 STMTHEAP 10000
;

```

run_dbmcfg.ddl

```

UPDATE DBM CFG USING
HEALTH_MON OFF
SHEAPTHRES 500000
MAX_QUERYDEGREE ANY
INTRA_PARALLEL NO
FCM_NUM_BUFFERS 16384
FCM_NUM_RQB 4096
NUM_POOLAGENTS 64
NUM_INITAGENTS 4
JAVA_HEAP_SZ 2048
CONN_ELAPSE 20
DFT_MON_TIMESTAMP OFF
CPUSPEED 2.715979e-07
;

```

change_logs.ksh

```

#!/usr/bin/ksh

db2_all "<<<+0< db2 update db cfg for tpcd using
newlogpath /dev/raw/raw61"
db2_all "<<<+1< db2 update db cfg for tpcd using
newlogpath /dev/raw/raw62"

```



```

db2_all "<<<+2< db2 update db cfg for tpcd using
newlogpath /dev/raw/raw61"
db2_all "<<<+3< db2 update db cfg for tpcd using
newlogpath /dev/raw/raw62"
db2_all "<<<+4< db2 update db cfg for tpcd using
newlogpath /dev/raw/raw61"
db2_all "<<<+5< db2 update db cfg for tpcd using
newlogpath /dev/raw/raw62"
db2_all "<<<+6< db2 update db cfg for tpcd using
newlogpath /dev/raw/raw61"
db2_all "<<<+7< db2 update db cfg for tpcd using
newlogpath /dev/raw/raw62"
db2_all "<<<+8< db2 update db cfg for tpcd using
newlogpath /dev/raw/raw61"
db2_all "<<<+9< db2 update db cfg for tpcd using
newlogpath /dev/raw/raw62"
db2_all "<<<+10< db2 update db cfg for tpcd using
newlogpath /dev/raw/raw61"
db2_all "<<<+11< db2 update db cfg for tpcd using
newlogpath /dev/raw/raw62"
db2_all "<<<+12< db2 update db cfg for tpcd using
newlogpath /dev/raw/raw61"
db2_all "<<<+13< db2 update db cfg for tpcd using
newlogpath /dev/raw/raw62"
db2_all "<<<+14< db2 update db cfg for tpcd using
newlogpath /dev/raw/raw61"
db2_all "<<<+15< db2 update db cfg for tpcd using
newlogpath /dev/raw/raw62"

```

tpcd.setup

```

# NOTE: ALL variable definitions must have a
comment at the end - haven't got
# the getvars script recognizing the
uncommented line yet
TPCD_PLATFORM=linux          # aix, nt, sun
....
TPCD_VERSION=2                # 1 or 2
(Version of tpcd). Default 1
TPCD_DBNAME=TPCD              # name to
create database under
TPCD_WORKLOAD=H                # TPC
version (R for TPCR, H for TPCH)
TPCD_AUDIT_DIR=/home/tpch/tpcd # top
level directory of tar file for
# all the tpcd scripts
TPCD_PRODUCT=v5                # v5 or pe
# Use pe if you really are
using pe v1.2!
# but I won't guarantee that
it will work!
TPCD_MODE=mpp                  #
uni/smp/mln/mpp
TPCD_PHYS_NODE=8               # number of
physical nodes

```

```

TPCD_LN_PER_PN=2              # number of
logical nodes per physical node
TPCD_SF=300                    # size of the
database (1=1GB,...) to
# get test size databases use:
# 0.012 = 12MB
# 0.1 = 100MB
TPCD_BUILD_STAGE=ALL          # where
to start the build - currently the
# following is possible:
# ALL - do everything
(create,load,
# index,stats,config)
(Default)
# CRTTBSP - start after
create db and
# config setting. Start
right at
# create tbsp
# LOAD - start from the
load of the tables
# INDEX - start from the
index creation
# (NOTE if earlyindex
is specified,
# then this will do the
create index
# followed by the
load...)
# RUNSTATS - start from
the runstats
# (NOTE Do not use
this option if
# distribution stats are
gathered
# as part of the load,
this will
# start after the load and
indices
# have been created.
# CONFIG - start from the
setting up of
# the benchmark runs
config setup
#
TPCD_DBPATH=/data/dbpath      # path for
database (defaults to home)
TPCD_DDLPATH=/home/tpch/custom # path
for all ddl files and customized
# scripts (load script),
config files,etc
TPCD_BUFFERPOOL_DEF=create_bufferpools.dd
l # name of file with bufferpool definitions
# and sizes
TPCD_NODEGROUP_DEF=create_nodegroups.dd
l # name of file in ddlpath with nodegroup

```

<pre> # definitions TPCD_EXPLAIN_DDL=NULL # file with DDL for explains statments # if this is NULL then uses the default # and puts it in USERSPACE1 across all # nodes...nt 1TB found it was faster if # just in a single node nodegroup TPCD_TBSP_DDL=create_tablespace.ddl # ddl file for tablespaces TPCD_DDL=create_tables.ddl # ddl file for tables TPCD_QUAL_TBSP_DDL=create_tablespace_qua l.ddl # ddl file for tablespaces for qual TPCD_QUAL_DDL=create_tables.ddl # ddl file for qualification database # tablespaces and tables should be identical # to regular ddl except container names TPCD_INDEXDDL=create_indexes.ddl # ddl file for indexes TPCD_EXTRAINDEX=no # no = no extra indexes # filename = If you want to create some # indices before # the load, and some indices after, then # use this additional file to specify the TPCD_ADD_RI=NULL # file name that contains any RI # constraints to add after index creation # set to NULL (default) if unused # indices to create after the load. TPCD_AST=NULL # file name that contains complete AST # definition including connection to # the database, summary table creation, # population, indexing and runstats. TPCD_RUNSTATS=runstats.ddl # ddl file for runstats. If you have # created indices before the load (ie # TPCD_EARLYINDEX=yes), have specified to </pre>	<pre> # gather stats on the load command (either # through your own load script or by using # TPCD_LOADSTATS=yes, AND you have # specified a file for TPCD_EXTRAINDEX # then this runstats file should include # the runstats commands specifically for # the extra indices. TPCD_RUNSTATSHORT=NULL # NOTE!! THIS IS BUGGY....I can't get it to # work on UNI successfully # ddl file for short runstats that are # run in the background while the # TPCD_RUNSTATS are run in the foreground # of the build. If this is used, then # TPCD_RUNSTATS should have the runstats # command for lineitem and # TPCD_RUNSTATSHORT should have runstats # commands for all other tables. TPCD_DBGGEN=/home/tpch/tpcd/appendix.v2/dbge n # path name to data generation code # Parameters used to specify source of # data for load scripts TPCD_INPUT=/data # NULL - use dbgen generated data OR # path name - to the pre- generated # flat files # TPCD_QUAL_INPUT=NULL # NULL - use dbgen generated data OR # path name - to the pre- generated # flat files TPCD_TAILOR_DIR=/home/tpch/tpcd/tailor # path name for the directory used to # generate split specific config files # only used for partitioned environment </pre>
--	--

TPCD_EARLYINDEX=no	# create indexes before the load		# the load will chose at run time
			# used in load script only
	# LOAD specific parameters follow:		TPCD_COPY_DIR=NULL # directory where copy image is created
TPCD_LOAD_DB2SET_SCRIPT=NULL	# Script that contains the db2set commands		# on load command
	# for the load process Use NULL if not		CURRENTLY UNUSED # used in load script only
	# specified		TPCD_FASTPARSE=yes # use fastparse on load
TPCD_LOAD_CONFIGFILE=load_dbcfg.ddl	#config file with specific database config		# used in load script only
	# parms for the load/index/runstats part		# Backup and logfile specific parameters follow:
	# of the build.		TPCD_BACKUP_DIR=/home/tpch/tpcd/backupdir
	# set to NULL if use defaults		# directory where backup files are placed
TPCD_LOAD_DBM_CONFIGFILE=load_dbmcfg.ddl	# config file with specific		TPCD_LOGPRIMARY=NULL # NULL/value = how many primary log files
	# database manager config parts for the		# to configure. If NULL is specified then
	# load/index/runstats part of the build.		# the default is not changed.
	# set to NULL if use defaults		TPCD_LOGFILSIZ=NULL # NULL/value = how 4KB pages to use for
TPCD_LOAD_QUALCONFIGFILE=load_dbcfg.d	# config file with specific database config		# logfilsiz db cfg parameter. If NULL is
	# parms for the load/index/runstats part		# specified then the default is not changed
	# of the build for qualification db.		TPCD_LOGSECOND=NULL # NULL/value = how many secondary log files
	# set to NULL if use defaults		# to configure. If NULL is specified then
TPCD_LOAD_DBM_QUALCONFIGFILE=load_d	# config file with specific		# the default is not changed.
bmcfg.ddl	# database manager config parts for the		TPCD_LOG_DIR=NULL # directory where log files stored..
	# load/index/runstats part of the build.		# NULL leaves them in the dbpath
	# set to NULL if use defaults		TPCD_LOG_QUAL_DIR=/home/logs # directory where qual log files stored
TPCD_LOADSTATS=no	# gather statistics during load		# NULL leaves them in the dbpath
	# ignored if EARLYINDEX is not set		TPCD_LOG=no # yes/no - whether to turn LOG_RETAIN on
	# due to runstats limitation		# i.e. are backups needed to be taken
TPCD_TEMP=/tmp/tpch	# path for LOAD temp files		# CONFIG specific parameters
	# used in load script only		TPCD_DB2SET_SCRIPT=run_db2set.ksh # Script that contains the db2set commands
TPCD_SORTBUF=4096	# sortbuf size for LOAD		# for the benchmark run.
	# used in load script only		Use NULL if not
TPCD_LOAD_PARALLELISM=0	# degree of parallelism to use on load		# specified
	# 0 = use the "intelligent default" that		TPCD_CONFIGFILE=run_dbcfg.ddl # name of configuration file in ddl path

```

# that will be used for the
benchmark run
TPCD_DBM_CONFIG=run_dbmcfg.ddl #
name of config file for database manager
# cfg parms
TPCD_QUALCONFIGFILE=run_dbcfg.ddl #
name of database cfg file in ddl path
# for qualification database
TPCD_DBM_QUALCONFIG=run_dbmcfg.ddl #
name of config file for database
# manager cfg parms

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

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

TPCD_ACTIVATE=no # activate the
database upon build
# completion

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

TPCD_SHARED_TEMP_FULL_PATHNAME=/home/tpch/sql/lib/tmp # just added
TPCD_QUERY_TEMPLATE_DIR=standard.V2
# subdirectory in AUDIT_DIR/queries
# to use as the source of the
query
# templates. Currently there
are
# v2 ones and pe ones. You
can make
# your own directory
following the same
# form as in the v2 directory
using
# any variant you wish
TPCD_QUAL_DBNAME=TPCDQUAL #
name of qualification database
TPCD_NUMSTREAM=6 # number of
streams for the throughput test
TPCD_FLATFILES=/home/ufdata # where
to generate/read flat files
# for update functions
TPCD_STAGING_TABLE_DDL=create_UFtables.
ddl # script that contains the ddl for creating
# the staging tables if
they are used for
# the update functions
TPCD_PRELOAD_STAGING_TABLE_SCRIPT=reloadUF.ksh # file that contains the sql for
preloading
# and gathering stats
on sample UF data
# Note that the data
used is sample data
# and is not data
from any of the applied
# update pairs
TPCD_DELETE_STAGING_TABLE_SQL=remove_UFtables.ddl # file that contains the sql for
deleting
# the preloaded
data from the staging
# tables
TPCD_UPDATE_IMPORT=false # true =
use import for the staging tables
# for UNI/SMP mode only
(mode change in
# tpcdbatch) (if not uni
mode then must
# change load_update)
# false = use load for
staging tables
# The default is false if not
set.
# NOTE that this parm is
only for UNI/SMP

```

invocation	# it is not for multi node	throughput starting at	# query stream of
			# different nodes or from
TPCD_SPLIT_UPDATES=64	# number	same node	
of chunks to split the update		TPCD_STATS_INTERVAL=30	# timing
	# function into.	interval for vmstats/iostats	
TPCD_CONCURRENT_INSERTS=8	#	TPCD_STATS_THRU_INT=300	# timing
number of insert chunks that are run		interval for vmstats/iostats for	
	# concurrently.		# throughput run
TPCD_SPLIT_UPDATES		TPCD_GATHER_STATS=off	# on/off -
	# should be evenly divisible	only implement for AIX yet	
by this number			# on = gather statistics
TPCD_CONCURRENT_INSERTS_LOAD=4		around power	
# number of insert chunks that are loaded			# test run
	# concurrently.	(vmstat,iostat,netstat)	
TPCD_SPLIT_UPDATES should			# off = no stats gathered
	# be evenly divisible by this	during power run	
number.			
	# this controls the load	TPCD_UFTEMP=DATA_INDEX	# base
portion of the		name of tablespace(s) where the	
	# insert routine for		# staging tables for the
partitioned databases		update functions	
TPCD_CONCURRENT_DELETES=8	#		# are created
number of delete chunks that are run			# this name will be used as
TPCD_SPLIT_DELETES=64	# number	the	
of portions to split the delete			# basename for the
	# function into.	tablespaces...eg	
	# this variable is only valid		# UFTEMP1 UFTEMP2
in UNI/SMP		TPCD_HAVECOMPILER=yes	# rebuild
	# mode.	tpcdbatch executable	
TPCD_GEN_UPDATEPAIRS=21	#		# yes/no
number of pairs of update function data		TPCD_SLEEP=5	# ?
	# to generate	TPCD_INLISTMAX=default	# max num
	# if 0 the update data	of keys to delete at a time	
generation and			# for UF2, use "default" for
	# setup will not be done.	default.	
use this if		TPCD_LOAD_SCRIPT=load_tables.ksh	# script
	# you don't want to run the	to run for loading tables	
update			# in TPCD_DDLPATH
	# functions (Update	directory under mln/mpp	
functions not			# leave as NULL if using
	# fully tested in new env't	default genloaduni	
yet)		TPCD_LOAD_SCRIPT_QUAL=NULL	#
TPCD_GENERATE_SEED_FILE=yes	#	script to run for loading tables in	
yes/no These are the seed files for			# TPCD_DDLPATH
	# generating the query	directory under mln/mpp	
substitution values			# for QUAL db
	# yes - generate a seed file	TPCD_ROOTPRIV=no	# do you have
base on		root privileges to be able	
	# year/month/day (for		# get values of things like
audited runs)		schedtune	
	# no - use qgen's default		# and vmtune (currently on
seeds		AIX only)	
TPCD_RUN_ON_MULTIPLE_NODES=NULL			# acid test specific
# pe V1.2 only - will we be running each		information	

```

TPCD_DB2LOG=/home/tpch/sql/lib/db2dump
# directory wehre the db2diag.log can
# be found for the durability
tests
TPCD_APPEND_ON=no # set to no if
the cluster indexes are used

```

Appendix C: Qualification Query Output

Qualification Queries

Query 1

Start timestamp 07/16/03 21:53:47.750000

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

Tag: Q1 Stream: -1 Sequence number: 17

```

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

```

```

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

```

Number of rows retrieved is: 4

Stop timestamp 07/16/03 21:53:50.830000
Query Time = 3.1 secs

Query 2

Start timestamp 07/16/03 21:52:37.100000

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

Tag: Q2 Stream: -1 Sequence number: 2

```

select
s_acctbal,
s_name,
n_name,
p_partkey,
p_mfgr,
s_address,

```

```

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

S_ACCTBAL      S_NAME
N_NAME         P_PARTKEY  P_MFGR
S_ADDRESS      S_PHONE
S_COMMENT
-----
-----
-----
-----
-----
9938.530 Supplier#000005359
UNITED KINGDOM      185358
Manufacturer#4
bgxj2K0w1kJvxY15mhCfou,W      33-429-
790-6131 blithely silent pinto beans are furiously.
slyly final deposits across
9937.840 Supplier#000005969
ROMANIA      108438 Manufacturer#1
rdnmd9c8EG1EIAYY3LPVa4yUNx6OwyVaQ

```

```

29-520-692-3537 carefully slow deposits use
furiously. slyly ironic platelets above the ironic
9936.220 Supplier#000005250
UNITED KINGDOM      249
Manufacturer#4      qX AB0vP8mJEWebUY9jri
33-320-228-2957 blithely special packages are.
stealthily express deposits across the closely final
instructi
9923.770 Supplier#000002324
GERMANY      29821 Manufacturer#4
uXcnR7tv87dG      17-779-299-1839
quickly express packages breach quiet pinto beans.
requ
9871.220 Supplier#000006373
GERMANY      43868 Manufacturer#5
iSLO35z7Ae      17-813-485-8637
never silent deposits integrate furiously blit
9870.780 Supplier#000001286
GERMANY      81285 Manufacturer#2
3gq0mZLHI5OTM6 tBYmLTHZaulCYnIECzQ7nj
17-516-924-4574 final theodolites cajole slyly
special,
9870.780 Supplier#000001286
GERMANY      181285
Manufacturer#4      3gq0mZLHI5OTM6
tBYmLTHZaulCYnIECzQ7nj 17-516-924-4574
final theodolites cajole slyly special,
9852.520 Supplier#000008973
RUSSIA      18972 Manufacturer#2
zVfUT3Np22kUC05tYWHBotaR      32-
188-594-7038 quickly regular instructions wake--
carefully unusual braids into the expres
9847.830 Supplier#000008097
RUSSIA      130557 Manufacturer#2
veMRTQBmUResNvfd3      32-375-
640-3593 slyly regular dependencies sleep slyly
furiously express dep
9847.570 Supplier#000006345
FRANCE      86344 Manufacturer#1
68yX tGXAkVRSxUGNSjJdptw 8O878xaFnaoQK
16-886-766-7945 silent pinto beans should have to
snooze carefully along the final reques
9847.570 Supplier#000006345
FRANCE      173827 Manufacturer#2
68yX tGXAkVRSxUGNSjJdptw 8O878xaFnaoQK
16-886-766-7945 silent pinto beans should have to
snooze carefully along the final reques
9836.930 Supplier#000007342
RUSSIA      4841 Manufacturer#4
icFgTpZ0TuAm188dv      32-399-414-
5385 final accounts haggle. bold accounts are
furiously dugouts. furiously silent asymptotes are
slyly
9817.100 Supplier#000002352
RUSSIA      124815 Manufacturer#2
XfLcJ71HKHnPgqvs7KNgPKcOWoWxo2w

```

32-551-831-1437 blithely pending packages across the ironic accounts grow slyly after the furiously 9817.100 Supplier#000002352	9492.790 Supplier#000005975
RUSSIA 152351 Manufacturer#3 XfLCj71HKHnPqgvs7KNgPKcOWoWxo2w	GERMANY 25974 Manufacturer#5 9UEiIp7uSYtTF5 17-992-579-4839
32-551-831-1437 blithely pending packages across the ironic accounts grow slyly after the furiously 9739.860 Supplier#000003384	always pending packages boost slyly. 9461.050 Supplier#000002536
FRANCE 138357 Manufacturer#2 D01XwXbcILNwmrGS6ZPrVhZxO40i 16- 494-913-5925 slyly ironic theodolites hag 9721.950 Supplier#000008757	UNITED KINGDOM 20033 Manufacturer#1 TEEkPusQ6rU18YvixE7IQtBDOyRBdGoOW12r 33-556-973-5522 even foxes are quickly furiously express requests. packages 9453.010 Supplier#000000802
UNITED KINGDOM 156241 Manufacturer#3 ryKUKeEWN7Z 33-821-407-2995 ironic, even dolphins above the furiously ironic foxes sleep slyly around the caref 9681.330 Supplier#000008406	ROMANIA 175767 Manufacturer#1 1Uj23QWxQjj7EyeqHWqGWTbN 29- 342-882-6463 final, regular packages across the slowly regular packag 9408.650 Supplier#000007772
RUSSIA 78405 Manufacturer#1 1A6x3PLy6F 32-139-873-8571 furiously even deposits affix thinly special theodolites. furiou 9643.550 Supplier#000005148	UNITED KINGDOM 117771 Manufacturer#4 rIoV2rj0KNy,IT 33-152-491-1126 blithely final ideas sleep carefully. requests are 9359.610 Supplier#000004856
ROMANIA 107617 Manufacturer#1 H7WOI6lzFuSsWzTSBrhzTYV 29-252- 617-4850 doggedly even ideas boost furiously against the furiously express 9624.820 Supplier#000001816	ROMANIA 62349 Manufacturer#5 k2CKOmXhPruJZ 29-334-870- 9731 carefully unusual packages sleep carefully even ideas. dogged accoun 9357.450 Supplier#000006188
FRANCE 34306 Manufacturer#3 NTwQPSZwfhc4uu1EMvEDopBnEv2j P 16- 392-237-6726 blithely regular accounts cajole furiously. regular 9624.780 Supplier#000009658	UNITED KINGDOM 138648 Manufacturer#1 LS,Z0 zbSvC7GWjF 33-583-607-1633 carefully regular deposits wake carefully furiously even i 9352.040 Supplier#000003439
ROMANIA 189657 Manufacturer#1 DmRxpLmL88XCBI0NB3tq3e0u 29- 748-876-2014 regular deposits haggle. furiously express asympto 9612.940 Supplier#000003228	GERMANY 170921 Manufacturer#4 B2bnKDlPkJp2uHKp 17-128-996-4650 fluffily regular pinto beans wake. unusual, final ideas c 9312.970 Supplier#000007807
ROMANIA 120715 Manufacturer#2 hnNBdhdXO4yT18 QNABTrL8fuv0A4p 29- 325-784-8187 carefully pending accounts serve. furiously close deposits boost slyly. q 9612.940 Supplier#000003228	RUSSIA 90279 Manufacturer#5 Dk2ebpGR3jlpYbpMg9Djr 32-673- 872-5854 unusual asymptotes above the 9312.970 Supplier#000007807
ROMANIA 198189 Manufacturer#4 hnNBdhdXO4yT18 QNABTrL8fuv0A4p 29- 325-784-8187 carefully pending accounts serve. furiously close deposits boost slyly. q 9571.830 Supplier#000004305	RUSSIA 100276 Manufacturer#5 Dk2ebpGR3jlpYbpMg9Djr 32-673- 872-5854 unusual asymptotes above the 9280.270 Supplier#000007194
ROMANIA 179270 Manufacturer#2 Bdj1T5EostLveb9ocRbz 29-973-481- 1831 furiously final deposits 9558.100 Supplier#000003532	ROMANIA 47193 Manufacturer#3 tJ96aHp8 l3uiq38LiDHswtk9bHMg 29-318- 454-2133 slyly ironic requests despite the unusual ins 9274.800 Supplier#000008854
UNITED KINGDOM 88515 Manufacturer#4 ncMxIJcDYzd5B7FIKxxLmnlzPeZB,FKBuJB 33-152-301-2164 daring, sly accounts breach about th	RUSSIA 76346 Manufacturer#3 ,uJfCd6eTiYE1ZEhDM vsc8ANQPWaPIQ 32- 524-148-5221 ruthlessly ironic instructions along the regular, furious requests integrate car 9249.350 Supplier#000003973
	FRANCE 26466 Manufacturer#1

OZSkIozfU4FYizk4e091MZHozL1qcHe257J89bW 16-722-866-1658 quickly ironic sauternes use b 9249.350 Supplier#000003973	578-432-2146 furiously final ideas believe furiously. furiously final ideas 8936.820 Supplier#000007043
FRANCE 33972 Manufacturer#1 OZSkIozfU4FYizk4e091MZHozL1qcHe257J89bW 16-722-866-1658 quickly ironic sauternes use b 9208.700 Supplier#000007769	UNITED KINGDOM 109512 Manufacturer#1 m5QHON1iD1OPhmU2R3z97u 6mCIvjnAc3i0,9s 33-784-177-8208 furiously regular excuses wake after the blithely special pinto beans? even instructions sl 8929.420 Supplier#000008770
ROMANIA 40256 Manufacturer#5 AzIENtMrVCSbrjyUu8 29-964-424- 9649 furiously ruthless epitaphs among the furiously regular accounts use slowly fluffily ev 9201.470 Supplier#000009690	FRANCE 173735 Manufacturer#4 aTOkYV7y3 kqbRrkOGJLaI 16-242- 746-9248 final accounts sleep furiously. blithely ironic foxes wake boldly across the furiously s 8920.590 Supplier#000003967
UNITED KINGDOM 67183 Manufacturer#5 pprpD77FEIWsNMmGT9T 33-121-267-9529 blithely unusual accounts integrate slyly. platelets 9192.100 Supplier#000000115	ROMANIA 26460 Manufacturer#1 NjCq3NUY82S 29-194-731-3944 quickly even requests should have to affix blithely-- fur 8920.590 Supplier#000003967
UNITED KINGDOM 85098 Manufacturer#3 EhrYy0MT5M1vfZ0V4skpifdp6pgFz5 33- 597-248-1220 slyly bold pinto beans boost across the furiously regular packages. carefully regu 9189.980 Supplier#000001226	ROMANIA 173966 Manufacturer#2 NjCq3NUY82S 29-194-731-3944 quickly even requests should have to affix blithely-- fur 8913.960 Supplier#000004603
GERMANY 21225 Manufacturer#4 BzfoA9wft1Mx3iBIs 17-725-903- 1381 final, express instruction 9128.970 Supplier#000004311	UNITED KINGDOM 137063 Manufacturer#2 d6sFwf6 TD1xyfuFbdM2h8LX7ZWc3zHupV 33-789- 255-7342 slyly ironic packages detect furious accounts. ironic de 8877.820 Supplier#000007967
RUSSIA 146768 Manufacturer#5 jSiHD4NTd8i9zVRX9uz9a, 32-155- 440-7120 regular pinto beans sleep ca 9104.830 Supplier#000008520	FRANCE 167966 Manufacturer#5 rXBIZqq9eWEuU90B vICab6 16-442- 147-9345 final deposits after the silent deposits ha 8862.240 Supplier#000003323
GERMANY 150974 Manufacturer#4 aA95nLn,m9shRrPXZw9Y1X 17-728- 804-1793 deposits sleep carefully e 9101.000 Supplier#000005791	ROMANIA 73322 Manufacturer#3 5RrF2PzoRlwpAGXjyf 29-736-951- 3710 unusual, pending theodolites integrate furiously slyly even pinto beans. unusual sheaves sleep befor 8841.590 Supplier#000005750
ROMANIA 128254 Manufacturer#5 txPYsp50HJkbbAJ0bYieqHmZtirDUVOcmC4lk 29-549-251-5384 carefully ironic packages after the 9094.570 Supplier#000004582	ROMANIA 100729 Manufacturer#5 n uXFrKx,KVYIQjmRuV,yejWmLMdRjnk 29-344-502-5481 excuses after the blithely regular packages mold carefully deposits. regular a 8781.710 Supplier#000003121
RUSSIA 39575 Manufacturer#1 5p,3Gp8kX 1EDarE0JR5juHH Sq9jlxgKenM 32- 587-577-1351 asymptotes above the slyly even requests haggle furiously about the regular accounts 8996.870 Supplier#000004702	ROMANIA 13120 Manufacturer#5 wdA7CLuYXS22oQEmp0V,x0PHrXiPdI5Rpwv,ub 29-707-291-5144 packages are quickly after the final, even packages. furiously regular 8754.240 Supplier#000009407
FRANCE 102191 Manufacturer#5 T35OahYXQGC 16-811-269- 8946 stealthy requests haggle c 8996.140 Supplier#000009814	UNITED KINGDOM 179406 Manufacturer#4 pj9oPHQ4OLWp 33-903-970-9604 regular dependencies haggle across the carefully bold
ROMANIA 139813 Manufacturer#2 RL,cVCKSXFc6Win6EmtF415A22as8nG2fqEa 29-995-571-8781 ironic theodolites are evenly unusual requests-- pending pinto beans across the in 8968.420 Supplier#000010000	
ROMANIA 119999 Manufacturer#5 R7kfmzyoIfXlrbnqNwUUW3phJctocp0J 29-	

8691.060 Supplier#000004429
 UNITED KINGDOM 126892
 Manufacturer#2 H0paE V6JCrIZpYrzI0LgIP
 33-964-337-5038 quickly special foxes against the
 furiously silent platelets wake quickly after t
 8655.990 Supplier#000006330
 RUSSIA 193810 Manufacturer#2
 7CsFQnd ,tzgMYvVoMim514DrJcX8SaQMTcy
 32-561-198-3705 blithely even packages alongside
 8638.360 Supplier#000002920
 RUSSIA 75398 Manufacturer#1
 iMYQSQzSLXg 32-122-621-
 7549 express deposits wake. furiously silent
 requests wake carefully silent instru
 8638.360 Supplier#000002920
 RUSSIA 170402 Manufacturer#3
 iMYQSQzSLXg 32-122-621-
 7549 express deposits wake. furiously silent
 requests wake carefully silent instru
 8607.690 Supplier#000006003
 UNITED KINGDOM 76002
 Manufacturer#2
 njRvqoOmIxNDe,da,SsnweINv1VY2YatifmJq
 33-416-807-5206 always special foxes wake slyly
 bold, ironic accounts. ironic instructions affix
 carefull
 8569.520 Supplier#000005936
 RUSSIA 5935 Manufacturer#5
 I3Qd1VwvDm5hYGzg1hBHzKy,P3YQXq7
 32-644-251-7916 packages sleep furiously. special
 requests about the fluffily even accounts detect
 8564.120 Supplier#000000033
 GERMANY 110032
 Manufacturer#1
 LLMgB3vXW,0g,8nuv3qU3QZaEBZvU2qRLX9
 17-138-897-9374 ironic instructions are. special
 pearls above
 8553.820 Supplier#000003979
 ROMANIA 143978 Manufacturer#4
 qLE5JpqDoe3XHsBI6etWpd4zRsjsBNb9Tidi6
 29-124-646-4897 express, ironic pinto beans cajole
 around the express, even packages. qu
 8517.230 Supplier#000009529
 RUSSIA 37025 Manufacturer#5
 NWW9SDThqi9RIeOA 32-565-297-
 8775 furiously silent requests cajole furiously
 furiously ironic foxes. slyly express p
 8517.230 Supplier#000009529
 RUSSIA 59528 Manufacturer#2
 NWW9SDThqi9RIeOA 32-565-297-
 8775 furiously silent requests cajole furiously
 furiously ironic foxes. slyly express p
 8503.700 Supplier#000006830
 RUSSIA 44325 Manufacturer#4
 qoW4lp2961uQiKOK6rW8 32-147-

878-5069 quickly regular excuses detect evenly
 around
 8457.090 Supplier#000009456
 UNITED KINGDOM 19455
 Manufacturer#1 U8pJ1
 SKbZPhH7,bLWXX3pG 33-858-440-
 4349 carefully final accounts sleep blithely special
 foxes. slyly regular pinto beans alon
 8441.400 Supplier#000003817
 FRANCE 141302 Manufacturer#2
 K6XLsYufTS 16-339-356-5115
 blithely blithe ideas are
 8432.890 Supplier#000003990
 RUSSIA 191470 Manufacturer#1
 wMJppCZ9aPMuq2nr88TVfztvE gj95OG wdNUE
 32-839-509-9301 final requests along the blithely
 ironic packages kindle against the carefully fina
 8431.400 Supplier#000002675
 ROMANIA 5174 Manufacturer#1
 kh18ydxR9VekbcMLgJKPtpNtwAkYtJTv 29-
 474-643-1443 express, final deposits cajole
 carefully. stealthily unusual requests
 8407.040 Supplier#000005406
 RUSSIA 162889 Manufacturer#4
 ITrK2mV94SooV6 Igo 32-626-152-
 4621 quickly final sheaves boost. car
 8386.080 Supplier#000008518
 FRANCE 36014 Manufacturer#3
 ZHAsABq5MRP
 e5kc0DRD8za3xGdf763ChHmoOA45 16-618-780-
 7481 slyly ironic theodolites are slyly. dogged,
 pendin
 8376.520 Supplier#000005306
 UNITED KINGDOM 190267
 Manufacturer#5 SyS2SsaA8i
 CqnbzUdfNH07bVtt9uW,Cp6FLCkOR 33-632-
 514-7931 furiously even instructions integrate
 during the furiously regular re
 8348.740 Supplier#000008851
 FRANCE 66344 Manufacturer#4
 E4uITlvmPHKvZ 16-796-240-
 2472 ironic instructions nag slyly against the slyly
 even theodolites. requests alongside of
 8338.580 Supplier#000007269
 FRANCE 17268 Manufacturer#4
 2vJh8wqp6CJp,W0Y 16-267-277-
 4365 ruthlessly regular asymptotes a
 8328.460 Supplier#000001744
 ROMANIA 69237 Manufacturer#5
 DfCXL6UWAY1lgjQYB0AjE8T2sx0BzS
 29-330-728-5873 blithely silent excuses are slyly
 above the furiously even courts
 8307.930 Supplier#000003142
 GERMANY 18139 Manufacturer#1
 OAPFw6SNodrC kFi 17-595-447-

6026 theodolites sleep blithely carefully regular warhorses. slyly regular ins
8231.610 Supplier#000009558
RUSSIA 192000 Manufacturer#2
FONKME0t7ZJhnjn9VL5 32-762-137-5858 slyly regular theodolites sleep fluffily express depos
8152.610 Supplier#000002731
ROMANIA 15227 Manufacturer#4
sDFx3iox2Zzx 29-805-463-2030 gifts use. slyly silent ideas are carefully beneath the silent instructions. slyly sil
8109.090 Supplier#000009186
FRANCE 99185 Manufacturer#1
wKLCzA5bMuGRBm35tvQAGpen23L 16-668-570-1402 quickly pending requests are blithely along the ironic, final requests; instr
8102.620 Supplier#000003347
UNITED KINGDOM 18344 Manufacturer#5
Froy39Y8ZUJ 33-454-274-8532 packages grow special orbits. regular theodolites about the carefully pe
8046.070 Supplier#000008780
FRANCE 191222 Manufacturer#3
rOssxn,6gRDzHr0gu,hEK 16-473-215-6395 regular epitaphs integrate slyly.
8042.090 Supplier#000003245
RUSSIA 135705 Manufacturer#4
oJSiGLXRCDAPcfWot7LkwSQRCh63XNS2 32-836-132-8872 carefully regular instructions integrate blithely silent foxes. furiously express instructions hagg1
8042.090 Supplier#000003245
RUSSIA 150729 Manufacturer#1
oJSiGLXRCDAPcfWot7LkwSQRCh63XNS2 32-836-132-8872 carefully regular instructions integrate blithely silent foxes. furiously express instructions hagg1
7992.400 Supplier#000006108
FRANCE 118574 Manufacturer#1
TyptNE7nv6BLpL16WFX 16-974-998-8937 regular pinto beans are after
7980.650 Supplier#000001288
FRANCE 13784 Manufacturer#4
tm0TjL5b oE 16-646-464-8247 unusual pinto beans cajole furiously according t
7950.370 Supplier#000008101
GERMANY 33094 Manufacturer#5
HG2wfVixwCIhK7dlrigGR3an2LuSifDJH 17-627-663-8014 quickly regular requests are furiously. pending deposits wake
7937.930 Supplier#000009012
ROMANIA 83995 Manufacturer#2
J6I7sJj0mGYIWFv9KxD3fK O7tvNP 29-250-925-9690 blithely bold ideas haggle quickly final, regular request

7914.450 Supplier#000001013
RUSSIA 125988 Manufacturer#2
AI9ODzBzWgnny28PHBei5M2IUFDd9 32-194-698-3365 final, ironic theodolites alongside of the ironic
7912.910 Supplier#000004211
GERMANY 159180 Manufacturer#5
Zva95Dwj EY0w,XjgsL700Zb2 l3almck 17-266-947-7315 final requests integrate slyly above the silent, even
7912.910 Supplier#000004211
GERMANY 184210 Manufacturer#4
Zva95Dwj EY0w,XjgsL700Zb2 l3almck 17-266-947-7315 final requests integrate slyly above the silent, even
7894.560 Supplier#000007981
GERMANY 85472 Manufacturer#4
e8hRUxe9cqQM3b 17-963-404-3760 regular, even theodolites integrate carefully. bold, special theodolites are slyly fluffily iron
7887.080 Supplier#000009792
GERMANY 164759 Manufacturer#3
3YSi76M2 I8XGikO5YgSM81r5Z6A7VkJcys 17-988-938-4296 pending, ironic packages sleep among the carefully ironic accounts. quickly final accounts
7871.500 Supplier#000007206
RUSSIA 104695 Manufacturer#1
YvrLdpD 5ExhHmRWzK41tw4 32-432-452-7731 furiously dogged pinto beans cajole. bold, express notornis until the slyly pending
7852.450 Supplier#000005864
RUSSIA 8363 Manufacturer#4
5odLpc1M83KXJ00 32-454-883-3821 blithely regular deposits
7850.660 Supplier#000001518
UNITED KINGDOM 86501 Manufacturer#1
ddNQX3hJjgico 33-730-383-3892 furiously final accounts wake carefully idle requests. even dolphins wake acc
7843.520 Supplier#000006683
FRANCE 11680 Manufacturer#4
Z1,hkHIw,Z3,,Comv6kLxIiPJtoNt 16-464-517-8943 carefully bold accounts doub

Number of rows retrieved is: 100

Stop timestamp 07/16/03 21:52:37.930000
Query Time = 0.8 secs

Query 3

Start timestamp 07/16/03 21:53:38.280000

-- Query 03 - Var_0 Rev_01 - Shipping Priority Query

Tag: Q3 Stream: -1 Sequence number: 11

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

L_ORDERKEY REVENUE
O_ORDERDATE O_SHIPPRIORITY

-
2456423 406181.011 1995-03-05
0
3459808 405838.699 1995-03-04
0
492164 390324.061 1995-02-19
0
1188320 384537.936 1995-03-09
0
2435712 378673.056 1995-02-26
0
4878020 378376.795 1995-03-12
0
5521732 375153.922 1995-03-13
0
2628192 373133.309 1995-02-22
0

993600 371407.459 1995-03-05
0
2300070 367371.145 1995-03-13
0

Number of rows retrieved is: 10

Stop timestamp 07/16/03 21:53:39.870000
Query Time = 1.6 secs

Query 4

Start timestamp 07/16/03 21:53:43.690000

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

Tag: Q4 Stream: -1 Sequence number: 14

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

O_ORDERPRIORITY ORDER_COUNT

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

Number of rows retrieved is: 5

Stop timestamp 07/16/03 21:53:45.300000
Query Time = 1.6 secs

Query 5

Start timestamp 07/16/03 21:53:55.490000

-- Query 05 - Var_0 Rev_02 Local Supplier Volume Query

Tag: Q5 Stream: -1 Sequence number: 20

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

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

Number of rows retrieved is: 5

Stop timestamp 07/16/03 21:54:05.780000
Query Time = 10.3 secs

Query 6

Start timestamp 07/16/03 21:52:59.840000

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

Tag: Q6 Stream: -1 Sequence number: 5

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

REVENUE

123141078.228

Number of rows retrieved is: 1

Stop timestamp 07/16/03 21:53:00.130000
Query Time = 0.3 secs

Query 7

Start timestamp 07/16/03 21:54:05.780000

-- Query 07 - Var_0 Rev_01 - Volume Shipping Query

Tag: Q7 Stream: -1 Sequence number: 21

```
select
supp_nation,
cust_nation,
l_year,
sum(volume) as revenue
from
```

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

```
SUPP_NATION      CUST_NATION
L_YEAR    REVENUE
```

```
-----
FRANCE      GERMANY
1995      54639732.734
FRANCE      GERMANY
1996      54633083.308
GERMANY     FRANCE
1995      52531746.670
GERMANY     FRANCE
1996      52520549.022
```

```
Number of rows retrieved is: 4
```

```
-----
Stop timestamp 07/16/03 21:54:13.020000
```

```
Query Time = 7.2 secs
```

Query 8

```
Start timestamp 07/16/03 21:53:03.270000
```

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

```
Tag: Q8 Stream: -1 Sequence number: 8
```

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

```
O_YEAR    MKT_SHARE
```

```
-----
1995      0.034
```

1996	0.041	NATION	O_YEAR	SUM_PROFIT
Number of rows retrieved is: 2		ALGERIA	1998	31342867.235
-----		ALGERIA	1997	57138193.023
Stop timestamp 07/16/03 21:53:32.110000		ALGERIA	1996	56140140.133
Query Time = 28.8 secs		ALGERIA	1995	53051469.653
Query 9		ALGERIA	1994	53867582.129
Start timestamp 07/16/03 21:52:37.930000		ALGERIA	1993	54942718.132
-----		ALGERIA	1992	54628034.713
-- Query 09 - Var_0 Rev_01 - Product Type Profit		ARGENTINA	1998	30211185.708
Measure Query		ARGENTINA	1997	50805741.752
Tag: Q9 Stream: -1 Sequence number: 3		ARGENTINA	1996	51923746.576
select		ARGENTINA	1995	49298625.767
nation,		ARGENTINA	1994	50835610.110
o_year,		ARGENTINA	1993	51646079.177
sum(amount) as sum_profit		ARGENTINA	1992	50410314.995
from		BRAZIL	1998	27217924.383
(BRAZIL	1997	48378669.199
select		BRAZIL	1996	50482870.357
n_name as nation,		BRAZIL	1995	47623383.635
year(o_orderdate) as o_year,		BRAZIL	1994	47840165.726
l_extendedprice * (1 - l_discount) - ps_supplycost *		BRAZIL	1993	49054694.035
l_quantity as amount		BRAZIL	1992	48667639.084
from		CANADA	1998	30379833.769
tpcd.part,		CANADA	1997	50465052.311
tpcd.supplier,		CANADA	1996	52560501.390
tpcd.lineitem,		CANADA	1995	52375332.809
tpcd.partsupp,		CANADA	1994	52600364.659
tpcd.orders,		CANADA	1993	52644504.073
tpcd.nation		CANADA	1992	53932871.697
where		CHINA	1998	31075466.165
s_suppkey = l_suppkey		CHINA	1997	50551874.450
and ps_suppkey = l_suppkey		CHINA	1996	51039293.875
and ps_partkey = l_partkey		CHINA	1995	49287534.617
and p_partkey = l_partkey		CHINA	1994	50851090.067
and o_orderkey = l_orderkey				
and s_nationkey = n_nationkey				
and p_name like '%green%'				
) as profit				
group by				
nation,				
o_year				
order by				
nation,				
o_year desc				

CHINA	1993	54229629.833	INDONESIA	1996	
CHINA	1992	52400529.372	51653060.117		
EGYPT	1998	29054433.386	INDONESIA	1995	
EGYPT	1997	50627611.452	51508779.594		
EGYPT	1996	49542212.845	INDONESIA	1994	
EGYPT	1995	48311550.321	52817950.322		
EGYPT	1994	49790644.736	INDONESIA	1993	
EGYPT	1993	48904292.969	47959994.955		
EGYPT	1992	49434932.619	INDONESIA	1992	
ETHIOPIA	1998		51776605.032		
28040717.267			IRAN	1998	29065736.238
ETHIOPIA	1997		IRAN	1997	50042063.054
47455009.866			IRAN	1996	50926653.188
ETHIOPIA	1996		IRAN	1995	51249667.648
46491097.573			IRAN	1994	50337085.865
ETHIOPIA	1995		IRAN	1993	51730763.490
46804449.301			IRAN	1992	49955856.563
ETHIOPIA	1994		IRAQ	1998	31624551.002
48516143.917			IRAQ	1997	55121749.019
ETHIOPIA	1993		IRAQ	1996	55897663.794
46551891.563			IRAQ	1995	54815472.517
ETHIOPIA	1992		IRAQ	1994	54408516.127
44934648.643			IRAQ	1993	53633167.977
FRANCE	1998	32226407.839	IRAQ	1992	55891939.340
FRANCE	1997	47121485.860	JAPAN	1998	27934179.670
FRANCE	1996	47263135.496	JAPAN	1997	44517162.546
FRANCE	1995	47275997.571	JAPAN	1996	42545606.120
FRANCE	1994	47067209.331	JAPAN	1995	43749356.400
FRANCE	1993	51163370.106	JAPAN	1994	44840243.070
FRANCE	1992	47846235.331	JAPAN	1993	44660015.533
GERMANY	1998		JAPAN	1992	45410249.122
28624942.660			JORDAN	1998	26901488.578
GERMANY	1997		JORDAN	1997	45471878.410
49309074.877			JORDAN	1996	46794325.792
GERMANY	1996		JORDAN	1995	45178828.576
49918683.168			JORDAN	1994	45333636.508
GERMANY	1995		JORDAN	1993	47971496.098
52650718.724			JORDAN	1992	44717239.177
GERMANY	1994		KENYA	1998	28597614.337
50346900.423			KENYA	1997	47949733.727
GERMANY	1993		KENYA	1996	46886924.623
50991895.806			KENYA	1995	46072338.755
GERMANY	1992		KENYA	1994	45772061.171
48274126.099			KENYA	1993	46308728.234
INDIA	1998	29943144.354	KENYA	1992	47257780.841
INDIA	1997	50665453.231	MOROCCO	1998	
INDIA	1996	50283092.291	26732115.580		
INDIA	1995	50006774.645	MOROCCO	1997	
INDIA	1994	48995190.756	45637304.250		
INDIA	1993	50286902.853	MOROCCO	1996	
INDIA	1992	50850329.402	45558221.745		
INDONESIA	1998		MOROCCO	1995	
27672339.996			47851318.887		
INDONESIA	1997		MOROCCO	1994	
50512145.726			46272172.944		

MOROCCO	1993		SAUDI ARABIA	1993
46764326.182			52937508.882	
MOROCCO	1992		SAUDI ARABIA	1992
48122783.583			54843459.641	
MOZAMBIQUE	1998		UNITED KINGDOM	1998
30712392.011			28494874.004	
MOZAMBIQUE	1997		UNITED KINGDOM	1997
50316528.763			49381810.899	
MOZAMBIQUE	1996		UNITED KINGDOM	1996
51640320.251			51386853.960	
MOZAMBIQUE	1995		UNITED KINGDOM	1995
50693774.506			51509586.788	
MOZAMBIQUE	1994		UNITED KINGDOM	1994
49253277.626			48086499.711	
MOZAMBIQUE	1993		UNITED KINGDOM	1993
49153016.537			49166827.223	
MOZAMBIQUE	1992		UNITED KINGDOM	1992
48247551.850			49349122.083	
PERU	1998	29326102.320	UNITED STATES	1998
PERU	1997	49753780.395	25126238.946	
PERU	1996	50935170.293	UNITED STATES	1997
PERU	1995	53309883.407	50077306.419	
PERU	1994	50643531.797	UNITED STATES	1996
PERU	1993	51584622.002	48048649.470	
PERU	1992	47523899.055	UNITED STATES	1995
ROMANIA	1998		48809032.423	
30368667.400			UNITED STATES	1994
ROMANIA	1997		49296747.183	
50365683.853			UNITED STATES	1993
ROMANIA	1996		48029946.801	
49598999.015			UNITED STATES	1992
ROMANIA	1995		48671944.498	
47537642.870			VIETNAM	1998
ROMANIA	1994		30442736.059	
51455283.010			VIETNAM	1997
ROMANIA	1993		50309179.794	
50407136.892			VIETNAM	1996
ROMANIA	1992		50488161.410	
48185385.129			VIETNAM	1995
RUSSIA	1998	28322384.027	49658284.612	
RUSSIA	1997	50106685.182	VIETNAM	1994
RUSSIA	1996	51753342.430	50596057.261	
RUSSIA	1995	49215820.365	VIETNAM	1993
RUSSIA	1994	52205666.441	50953919.152	
RUSSIA	1993	51860230.034	VIETNAM	1992
RUSSIA	1992	53251677.153	49613838.315	
SAUDI ARABIA	1998			
31541259.810				
SAUDI ARABIA	1997		Number of rows retrieved is: 175	
52438750.808			-----	
SAUDI ARABIA	1996			
52543737.820				
SAUDI ARABIA	1995		Stop timestamp 07/16/03 21:52:57.780000	
52938696.533			Query Time = 19.9 secs	
SAUDI ARABIA	1994			
51389601.967				

Query 10

Start timestamp 07/16/03 21:53:50.830000

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

Tag: Q10 Stream: -1 Sequence number: 18

```

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

```

C_CUSTKEY	C_NAME	REVENUE
C_ACCTBAL	N_NAME	
C_ADDRESS	C_PHONE	
C_COMMENT		
57040	Customer#0000057040	
734235.246	632.870 JAPAN	

nICtsILWBB	22-895-641-3466	
requests sleep blithely about the furiously i		
143347	Customer#0000143347	
721002.695	2557.470 EGYPT	
,Q9MI3w0gvX	14-742-935-3718	
fluffily bold excuses haggle finally after the u		
60838	Customer#0000060838	
679127.308	2454.770 BRAZIL	
VWmQhWweqj5hFpcvhGfBeOY9hJ4m		
12-913-494-9813	furiously even pinto beans	
integrate under the ruthless foxes; ironic, even		
dolphins across the slyl		
101998	Customer#0000101998	
637029.567	3790.890 UNITED	
KINGDOM	0,ORojfDdyMca2E2H	
33-593-865-6378	accounts doze blithely! enticing,	
final deposits sleep blithely special accounts. slyly		
express accounts pla		
125341	Customer#0000125341	
633508.086	4983.510 GERMANY	
9YRcnoUPOM7Sa8xymhsDHdQg	17-	
582-695-5962	quickly express requests wake	
quickly blithely		
25501	Customer#0000025501	
620269.785	7725.040 ETHIOPIA	
sr4VVVe3xCJQ2oo2QEhi19D,pXqo6kOGaSn2		
15-874-808-6793	quickly special requests sleep	
evenly among the special deposits. special deposi		
115831	Customer#0000115831	
596423.867	5098.100 FRANCE	
AlMpPnmtGrOfrDMUs5VLo		
EIA,Cg,Rw5TBuBoKiO	16-715-386-3788	
carefully bold excuses sleep alongside of the thinly		
idle		
84223	Customer#0000084223	
594998.024	528.650 UNITED KINGDOM	
Eq51o UpQ4RBr fYTdrZApDsPV4pQyuPq		
33-442-824-8191	pending, final ideas haggle final	
requests. unusual, regular asymptotes affix		
according to the even foxes.		
54289	Customer#0000054289	
585603.392	5583.020 IRAN	
x3ouCpz6,pRNVhjr0CCQG1	20-834-	
292-4707	express requests sublata blithely regular	
requests. regular, even ideas solve.		
39922	Customer#0000039922	
584878.113	7321.110 GERMANY	
2KtWzW,FYkhdWBfobp6SFXWYKjvU9		
17-147-757-8036	even pinto beans haggle. slyly	
bold accounts inte		
6226	Customer#000006226	
576783.761	2230.090 UNITED	
KINGDOM		
TKbxS1dbkGMtaa,KOi26lbip4P0tPbWK0	33-	
657-701-3391	quickly final requests against the	

regular instructions wake blithely final instructions.
pa

922 Customer#000000922
 576767.533 3869.250 GERMANY
 rsR9lRxyTdHbDOvt8nYbwjK5vAWH9sB
 17-945-916-9648 boldly final requests cajole blith
 147946 Customer#0000147946
 576455.132 2030.130 ALGERIA
 Jqdt1kHAJtuTqHQK,B7 3tJh 10-886-
 956-3143 furiously even accounts are blithely
 above the furiousl
 115640 Customer#0000115640
 569341.193 6436.100 ARGENTINA
 6yKLIRRAirUmBjKNO6Z3 11-411-
 543-4901 final instructions are slyly according to
 the
 73606 Customer#0000073606
 568656.858 1785.670 JAPAN
 vx9,7ACVtoKnLcoAHGNYDF 22-437-
 653-6966 furiously bold orbits about the furiously
 busy requests wake across the furiously quiet
 theodolites. d
 110246 Customer#0000110246
 566842.981 7763.350 VIETNAM
 UgsLFL3rendATzcHi 31-943-426-
 9837 dolphins sleep blithely among the slyly final
 142549 Customer#0000142549
 563537.237 5085.990 INDONESIA
 pJAmChWXct HNjPzgoBUOgAHduwwIR
 19-955-562-2398 regular, unusual dependencies
 boost slyly; ironic attainments nag fluffily into the
 unusual packages?
 146149 Customer#0000146149
 557254.986 1791.550 ROMANIA
 STLwtlaB6 29-744-164-6487
 silent, unusual requests detect quickly slyly regul
 52528 Customer#0000052528
 556397.351 551.790 ARGENTINA
 elsy8c9Z,7ch 11-208-192-3205
 unusual requests detect. slyly dogged theodolites use
 slyly. deposit
 23431 Customer#0000023431
 554269.536 3381.860 ROMANIA
 kKI5,CJAJQjQRQtOdCiFQ 29-915-
 458-2654 instructions nag quickly. furiously bold
 accounts cajol

Number of rows retrieved is: 20

Stop timestamp 07/16/03 21:53:52.830000
Query Time = 2.0 secs

Query 11

Start timestamp 07/16/03 21:53:45.300000

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

Tag: Q11 Stream: -1 Sequence number: 15

```

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

```

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

Rows Deleted

128820	7892882.720
25891	7890511.200

```

122819      7888881.020
154731      7888301.330
101674      7879324.600
51968       7879102.210
72073       7877736.110
5182        7874521.730

```

Number of rows retrieved is: 1048

Stop timestamp 07/16/03 21:53:46.380000
Query Time = 1.1 secs

Query 12

Start timestamp 07/16/03 21:54:13.020000

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

Tag: Q12 Stream: -1 Sequence number: 22

```

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

```

```

L_SHIPMODE HIGH_LINE_COUNT
LOW_LINE_COUNT
-----

```

```

MAIL          6202      9324
SHIP          6200      9262

```

Number of rows retrieved is: 2

Stop timestamp 07/16/03 21:54:16.190000
Query Time = 3.2 secs

Query 13

Start timestamp 07/16/03 21:53:36.110000

-- Query 13 - Var_0 Rev_01 - Customer Distribution
Query

Tag: Q13 Stream: -1 Sequence number: 10

```

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

```

```

C_COUNT  CUSTDIST
-----
0        50004
9        6641
10       6566
11       6058
8        5949

```

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

Number of rows retrieved is: 42

Stop timestamp 07/16/03 21:53:38.280000
 Query Time = 2.2 secs

Query 14

Start timestamp 07/16/03 21:52:13.240000

 --#SET ROWS_OUT -1 ROWS_FETCH -1

-- Query 14 - Var_0 Rev_01 - Promotion Effect
 Query

Tag: Q14 Stream: -1 Sequence number: 1

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

PROMO_REVENUE

 16.381

Number of rows retrieved is: 1

Stop timestamp 07/16/03 21:52:37.100000
 Query Time = 23.9 secs

Query 15a

Start timestamp 07/16/03 21:53:46.380000

-- Query 15 - Var_a Rev_01 - Top Supplier Query

Tag: Q15a Stream: -1 Sequence number: 16

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

```

select
s_suppkey,
s_name,
s_address,
s_phone,
total_revenue
from
tpcd.supplier,
revenue
where
s_suppkey = supplier_no
and total_revenue = (
select
max(total_revenue)
from
revenue
)
order by
s_suppkey

```

```

S_SUPPKEY  S_NAME
S_ADDRESS          S_PHONE
TOTAL_REVENUE
-----

```

```

      8449 Supplier#000008449
5BXWsJERA2mP5OyO4          20-469-
856-8873      1772627.209

```

Number of rows retrieved is: 1

Stop timestamp 07/16/03 21:53:47.750000
Query Time = 1.4 secs

Query 16

Start timestamp 07/16/03 21:53:41.440000

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

Tag: Q16 Stream: -1 Sequence number: 13

```

select
p_brand,
p_type,
p_size,
count(distinct ps_suppkey) as supplier_cnt
from

```

```

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

```

```

P_BRAND  P_TYPE          P_SIZE
SUPPLIER_CNT
-----
---
Brand#41  MEDIUM BRUSHED TIN          3
28
Brand#54  STANDARD BRUSHED COPPER
14      27
Brand#11  STANDARD BRUSHED TIN
23      24
Brand#11  STANDARD BURNISHED BRASS
36      24
Brand#15  MEDIUM ANODIZED NICKEL
3       24
Brand#15  SMALL ANODIZED BRASS
45      24

```

Rows Deleted

```

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

```

Number of rows retrieved is: 18314

Stop timestamp 07/16/03 21:53:43.690000
Query Time = 2.3 secs

Query 17

Start timestamp 07/16/03 21:53:00.130000

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

Tag: Q17 Stream: -1 Sequence number: 6

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

AVG_YEARLY

348406.054

Number of rows retrieved is: 1

Stop timestamp 07/16/03 21:53:01.410000
Query Time = 1.3 secs

Query 18

Start timestamp 07/16/03 21:53:01.410000

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

Tag: Q18 Stream: -1 Sequence number: 7

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

C_NAME C_CUSTKEY
O_ORDERKEY O_ORDERDATE
O_TOTALPRICE 6

Customer#0000128120 128120 4722021
1994-04-07 544089.090 323.000
Customer#0000144617 144617 3043270
1997-02-12 530604.440 317.000
Customer#0000013940 13940 2232932
1997-04-13 522720.610 304.000
Customer#0000066790 66790 2199712
1996-09-30 515531.820 327.000

Customer#0000046435	46435	4745607	Customer#0000036619	36619	4806726
1997-07-03	508047.990	309.000	1995-01-17	446704.090	328.000
Customer#0000015272	15272	3883783	Customer#0000141823	141823	2806245
1993-07-28	500241.330	302.000	1996-12-29	446269.120	310.000
Customer#0000146608	146608	3342468	Customer#0000053029	53029	2662214
1994-06-12	499794.580	303.000	1993-08-13	446144.490	302.000
Customer#0000096103	96103	5984582	Customer#0000018188	18188	3037414
1992-03-16	494398.790	312.000	1995-01-25	443807.220	308.000
Customer#0000024341	24341	1474818	Customer#0000066533	66533	29158
1992-11-15	491348.260	302.000	1995-10-21	443576.500	305.000
Customer#0000137446	137446	5489475	Customer#0000037729	37729	4134341
1997-05-23	487763.250	311.000	1995-06-29	441082.970	309.000
Customer#0000107590	107590	4267751	Customer#0000003566	3566	2329187
1994-11-04	485141.380	301.000	1998-01-04	439803.360	304.000
Customer#0000050008	50008	2366755	Customer#0000045538	45538	4527553
1996-12-09	483891.260	302.000	1994-05-22	436275.310	305.000
Customer#0000015619	15619	3767271	Customer#0000081581	81581	4739650
1996-08-07	480083.960	318.000	1995-11-04	435405.900	305.000
Customer#0000077260	77260	1436544	Customer#0000119989	119989	1544643
1992-09-12	479499.430	307.000	1997-09-20	434568.250	320.000
Customer#0000109379	109379	5746311	Customer#0000003680	3680	3861123
1996-10-10	478064.110	302.000	1998-07-03	433525.970	301.000
Customer#0000054602	54602	5832321	Customer#0000113131	113131	967334
1997-02-09	471220.080	307.000	1995-12-15	432957.750	301.000
Customer#0000105995	105995	2096705	Customer#0000141098	141098	565574
1994-07-03	469692.580	307.000	1995-09-24	430986.690	301.000
Customer#0000148885	148885	2942469	Customer#0000093392	93392	5200102
1992-05-31	469630.440	313.000	1997-01-22	425487.510	304.000
Customer#0000114586	114586	551136	Customer#0000015631	15631	1845057
1993-05-19	469605.590	308.000	1994-05-12	419879.590	302.000
Customer#0000105260	105260	5296167	Customer#0000112987	112987	4439686
1996-09-06	469360.570	303.000	1996-09-17	418161.490	305.000
Customer#0000147197	147197	1263015	Customer#0000012599	12599	4259524
1997-02-02	467149.670	320.000	1998-02-12	415200.610	304.000
Customer#0000064483	64483	2745894	Customer#0000105410	105410	4478371
1996-07-04	466991.350	304.000	1996-03-05	412754.510	302.000
Customer#0000136573	136573	2761378	Customer#0000149842	149842	5156581
1996-05-31	461282.730	301.000	1994-05-30	411329.350	302.000
Customer#0000016384	16384	502886	Customer#0000010129	10129	5849444
1994-04-12	458378.920	312.000	1994-03-21	409129.850	309.000
Customer#0000117919	117919	2869152	Customer#0000069904	69904	1742403
1996-06-20	456815.920	317.000	1996-10-19	408513.000	305.000
Customer#0000012251	12251	735366	Customer#0000017746	17746	6882
1993-11-24	455107.260	309.000	1997-04-09	408446.930	303.000
Customer#0000120098	120098	1971680	Customer#0000013072	13072	1481925
1995-06-14	453451.230	308.000	1998-03-15	399195.470	301.000
Customer#0000066098	66098	5007490	Customer#0000082441	82441	857959
1992-08-07	453436.160	304.000	1994-02-07	382579.740	305.000
Customer#0000117076	117076	4290656	Customer#0000088703	88703	2995076
1997-02-05	449545.850	301.000	1994-01-30	363812.120	302.000
Customer#0000129379	129379	4720454			
1997-06-07	448665.790	303.000			
Customer#0000126865	126865	4702759			
1994-11-07	447606.650	320.000			
Customer#0000088876	88876	983201			
1993-12-30	446717.460	304.000			

Number of rows retrieved is: 57

Stop timestamp 07/16/03 21:53:03.270000
Query Time = 1.9 secs

Query 19

Start timestamp 07/16/03 21:53:52.830000

-- Query 19 - Var_0 Rev_01 - Discounted Revenue
Query

Tag: Q19 Stream: -1 Sequence number: 19

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

REVENUE

3083843.058

Number of rows retrieved is: 1

Stop timestamp 07/16/03 21:53:55.490000
Query Time = 2.7 secs

Query 20

Start timestamp 07/16/03 21:52:57.780000

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

Tag: Q20 Stream: -1 Sequence number: 4

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

)
 and s_nationkey = n_nationkey
 and n_name = 'CANADA'
 order by
 s_name

S_NAME	S_ADDRESS
Supplier#000000020	JtPqm19E7tF
152R11wQZ8j0H	
Supplier#000000091	
35WVnU7GLNbQDcc2TARavGtK6RB6ZCd46UAY	
Supplier#000000197	
3oYqODDUGH3XsHXmPuzYHW5NLU3,ONZI	
Supplier#000000226	
TXAcN7Ym29ObaqaKmyDri1mN	
Supplier#000000285	q TMZEDyZtv
vUiFKBhT3NjInIxpL	
Supplier#000000378	mLPJtpu4wOc cSFzBR
Supplier#000000402	
JR8vWoCteJtJg3okRpt0r28KEo	
Supplier#000000530	
0BvoewCPg2scOEFuL93FRKqSxHmdhw1	
Supplier#000000688	
orLvVDOBBE2B2ppjbiTZTJoIJgn1VgKg3	
Supplier#000000710	
MZR2bxcPrbC2vHho0CbuOBkLJMpqOwghC	
Supplier#000000736	
GUIYDfv5xCxLgDx6KQ8khY ntVVnFqmfMKIgt	
Supplier#000000761	tF8fMGa6HY4
w77mDwT4rO21kxwe7uTSYNW	
Supplier#000000884	QEJm8KMHQ8
Supplier#000000887	y mQ7NHjVbdqnbYr9 L
Supplier#000000935	
JHRSOterYgt4MTNo7cupTza,6MoNw 4	
Supplier#000000975	
IqorM1ypBdwgPVuf6sMCKuF9D1rJN1iCTXKmal	
St	
Supplier#000001263	Aa4
UELS1JqY7qIjniwCWic	
Supplier#000001399	gE COEie8t3c4xqeuT
Supplier#000001446	
GOJRjfd6Z9UQ,fiuPz6CO5GDGV	
Supplier#000001454	8cBIELxY754iHJIgRXJ0
Supplier#000001500	wnElVvfuyaPJROy6x0
Supplier#000001602	ygd4iNQLQeVWW
Supplier#000001626	7Jud9t6xZNzIEB,
Supplier#000001682	C37Gkv 7a5ujZ9
Supplier#000001699	bRoW
MhYUCJH5bABOA53N9i	
Supplier#000001700	UJeFoDLZ2VtflK
Supplier#000001726	
8M92T8y7jYXzmvCANTtqR8GHuT	

Supplier#000001730	aOSM0,btPDs
UsC06himJn,6nswJG	
Supplier#000001746	
B4zMDOFcMGbkZQ4XxA,UyaQoEWZzMSGI	
wTsJp9N	
Supplier#000001752	m
QTCyyuWZ8j2hQa7JNclIpGr	
Supplier#000001856	
I3O9Kt2l,QK,VI9TBuV9PhhfP6kt2J69bwmv64e	
Supplier#000001931	
mBhPe7YJU1aYmpwTiRdivl	
Supplier#000001939	e G0hrWPL9N6z
Supplier#000001990	
o8nhGpKphsybKZtye0o6OfDz9GIcuGHjMs7pq	
Supplier#000002020	IpU ZOTeubVOC
Supplier#000002022	
sNvMPI3TiNZBOYV3w2XX	
Supplier#000002036	Z,ty7z5cPHh66iY5op,q
Supplier#000002204	y2EF
XUo,UyNoAQEH,gIazC7aRG1zmuzzf	
Supplier#000002243	E8cm5YhMc6UR
Supplier#000002245	
KsYA4445HcugJAb3eCmvtUsIGA7Qne	
Supplier#000002282	
n8YZgSNuC4,iZ7s5oHTMHNfDv94DwZ2rrUEb0p	
gD	
Supplier#000002303	
EoC4LcP2cuEPKcKyTFyMGFBGkF	
Supplier#000002373	
asj8ud7aEmGoHuiqI5qVZ1rhpWS hJc9tF9	
Supplier#000002419	
BtNpaOZWiVGVE53RWL22	
Supplier#000002481	
Efgj6dDYeFS7cq,V1DARlmeOiGecY	
Supplier#000002571	i16xKt,WOrJhlf
GkzsRdrd04sZ 5jei9MtB	
Supplier#000002585	
pzbCgCvYax82Wq5,dG4xzyDMiRW8d	
Supplier#000002630	
2iaqwId62RhjVIwNrsETHr5FbJnFTopCYZ	
Supplier#000002719	
XDDsajYoPAama6xjz37p66GQbXEd8X	
Supplier#000002721	k5NlqeYhjeb8BgE
Supplier#000002730	
Gilu9XLsEX,oU0EyshvFTWs	
Supplier#000002775	unOFQoQpnWJJj
Supplier#000002853	A7dqcyj3GQ
Supplier#000002875	
VhKelsRayU6EvfXgyyZ8aFgvv0HMfa19q1Q	
Supplier#000002934	F0y pndtv8r vKXoJp
Supplier#000002941	eQNNPRrS27ngMG1ub,
Supplier#000002960	hobom
9kkCVwuS7uKjJU0Eq2LS 8ya8Y7	
Supplier#000002980	
qRGRt6ztQ3x4Gyc9jMsAkIqmBSIN,8hsGv	

Supplier#000003062	g8adKB2ZuDdssRsQoQBtUjvc1b	Supplier#000004430	uw8zge87fVkf3pm8WFxc bdfuhsqeG
Supplier#000003087	rdvMSbz1W	Supplier#000004522	v3yogwz1nqAuiipkoLs3YcKCPk2zpK
LHZj8B5sRR,M4a		Supplier#000004527	CrB53pDuKOHF4IUrZCXck2XA4V
Supplier#000003089	VPr5pi1Ae915QrArKTf8NOKYSkp7mU iZu	Supplier#000004542	eh8Pfin C2IYtZ Y nIgL
Supplier#000003095	kucdXIhJ6IYsHy0ArE7n	Supplier#000004574	,jwkvE5xM1WoiDNP
Supplier#000003201	nSTtv	Supplier#000004655	VTdApPWryXckXISRcrizB4dAWHkQbCw2f
Ui0y0BdzWb4T6snugIhEhn14yM		Supplier#000004701	VH3WxXT69sjeJL
Supplier#000003213	Cu 9bXI aZ6CtLa1N7LX	Supplier#000004711	QmsHB,aNaxsFzYmaeuw,HERwDVPxYs4FfZ
Supplier#000003241	I,U861f8RcYEvHqfc5IqeMiJP	Supplier#000004987	s6luZyBhSexcwKIIqS
Supplier#000003275	SucWDuhYahP3UwkM	Supplier#000005000	oM3S,Xrv,jTl
Supplier#000003288	nnNLdzTmV6P0uf7yBCiK3fWt2UxHJ0	Supplier#000005100	dLw2b
Supplier#000003313	nFYiTWM0,W8S8b	Supplier#000005192	inBW K3Inv0HLU9k
wDE6EWK1vhCdJ1Gwgufh3L5j		Supplier#000005195	5CIXPZ1QIOcJr18MxZmL4Jm
Supplier#000003314	IDIz6TesAcXI6pXtzbO	Supplier#000005283	WLu23vv3t08a30eBnoYJvsta
uzevXon6CH9WATfo		Supplier#000005300	L3kYS3ABu6
Supplier#000003380	Ibw,51BzsCyx17X3zqAFc0Gu7Yx0K7mImP9z0u	Supplier#000005386	7PUqqMjB4f5b
Supplier#000003403	NX3YCs0gokSyjHJSqr34oullZJ14pw	Supplier#000005426	SnsYc67ZAr
Supplier#000003421	9JXNyS4VCMDL14CxDIJ L0	PWpgUZFHha,3Hbp39bwc	
Supplier#000003441	twlhit80C6y8JjHCO3	Supplier#000005484	QINcarA4Trl20XRzbAIJqQZZfCYyBpCd,pzb gC
Supplier#000003590	tTRoffuAP1oPC	Supplier#000005505	OYzPHIWxgVfb3
Supplier#000003607	GdAljb2Hv8rGL	Supplier#000005506	dDrLVuBsC4Lp
Supplier#000003625	B2VSS5,2GVJQ6tZa37KdAmg erHp2jPgZ	Supplier#000005516	4zdRRrhZRvmxOCJ6VH9nYfMMPamLTUEoSvCw
Supplier#000003656	Nm2EEdYKEnshNLkYY3ynLh50NQVU	y2	
Supplier#000003782	J5zb1LTPG,V7AdewI,fgQf6 o,sE K	Supplier#000005536	esCRylGBKPk70mlXn0TTf2gFWRJZo
Supplier#000003918	FM9w9pzhCqQLAN,aMW	Supplier#000005605	U5HUmJF,E7KAGE
Supplier#000003941	cEP,VFaLpe9UZScU4gAlirRwtX	Supplier#000005631	,W75
Supplier#000003994	5,,f1CXdHg0	IFsCY9hmp2pnKBew7Fwv9Ao	
Supplier#000004005	6TYXl,votYNqWcKjxS7Hqyc56kB	Supplier#000005730	W
Supplier#000004033	EOzqJxRhM0HI37dILPY	GP,c8MvzsjuGgShQnXZ6BD9IYOKTjZ	
Supplier#000004140	0KfTnit2HOkf	Supplier#000005736	ZNAr382Jy258LB
Supplier#000004165	w7hYN2daqScYCIRSdUn7XTDNju03gnZ	Supplier#000005737	OEm4O9XYoXHuoN0qRVrcF2DWS
Supplier#000004207	zLUWBwIceWjG4HdYE80M0V4xqIfu	Supplier#000005797	1C0cMQv9P8oE5FRLdRLB5boIAqr6CKx8F89
Supplier#000004236	OF0jzbzhEkICu2z8BDRvEBGx4H y0EoNhScU8	Supplier#000005836	zuX8HcnY1x5klp9k0
Supplier#000004246	4JQrQ4alTxXA6Xf jn	Supplier#000005875	Gg0z2JkspRXJ8tjuRuw82IP5aEO1MYg31xkQ8
Supplier#000004278	QpNNPCpui5CrnIR	Supplier#000005974	amJ9VIm0Ffya3wMVW8v3t8Kz985IJy
Supplier#000004343	lgXzPDBA anG4fex5plol	Supplier#000005989	AHI20WGKfBqxTO
Supplier#000004346	9X,TUgmcvC	Supplier#000006059	XE,OwTevhRu3YwFwir1
Supplier#000004388	6L1rF,h0EvCWQ8	Supplier#000006065	7IiYouX4W7yVzfGsfwx3g9tUgJfKw
Supplier#000004406	rJ,1QgxU5vyLb46t0TGP4K21JRxmQjBkiK		

Supplier#000006070	8QFoVF,Bn	Supplier#000007561	AeOlKZVX,5p
dUn1PEKLEkFAM		Supplier#000007789	AaTO6Pb yxOtcX
Supplier#000006109		wd8GJEAjUB L47Z8s	
A2VKPMJXNgkDtFOb67bkpvDPM		Supplier#000007801	VRLI07Z UME6Pr
Supplier#000006121		Supplier#000007818	uJJOYpaMOri
9RYtO5vms23vWkzCophd,4exjKC1		wOStApISY	
Supplier#000006215		Supplier#000007885	
IYImP8zF0Vc4NA42TGZtJi8PTByIIIGHnibnC		yXzIOPJJV1Ct76BeZOheOqCQQi4K2	
Supplier#000006217		Supplier#000007918	A0wREqaUfCn2tHZ
a5dYX927RHND6MQ5k36N		Supplier#000007926	n solT,gR6u
Supplier#000006274		Supplier#000007957	
9Xt715Au7g7ArKrbaKO4SrqAJpGd1z5VZJjyv6		nfvDIDZWzzD6ZrNat1ifscgX5zsP	
Supplier#000006435		Supplier#000007965	mT6DVGhTBVKJH
viKmUS3zs2QDcWmDDTOjkcSt		Supplier#000007968	ozlR6G1Y1CVj3dRRm
Supplier#000006463		wtLZIGjDnVSYk8k1Bc	
UrvGNIYmcWSICyFNtYGEjerqnCf8zsl5X9d5H		Supplier#000007998	
Supplier#000006493	DH5rL1zdQpVjET	gDq8lqL29ldCRNUO0Qzpx5ARfDYb	
1LGDn67fUXAQhKqI1g		Supplier#000008168	RcQSYRS1PgpDLnf4
Supplier#000006521	QRrYsIjsu9	Supplier#000008231	jgTMkwr2HR0 7NB
Supplier#000006607	YlrYMZKANvYvP	b0wOB4ufp	
Supplier#000006706		Supplier#000008243	
rGWKQ1MbxZb1UoXBG		ZqtMbfGnAEt5sHk8Is3yKlfCSKrmIxOeucFiik	
AHDzQ3,LuwA8SwGPM		Supplier#000008275	qFPdnM4K0KC3gaFgd
Supplier#000006761	EWJuleApVC	Supplier#000008323	UViZS
nZjKBfwwA48ycgFFQ		1Eq8wErbcnJM9eOHRyECtMa0qLo3dWpiqP	
Supplier#000006808		Supplier#000008366	
kkNY3DrRDmPjhJ1x3H3u5giBqC7		KTTSONHZWpy4RcmhFwb75AWIvr89Umqp3dTt	
Supplier#000006858		M8S	
MDFid8SSVwqpJz4w7k110DYyYkVvk2ZVJrkjiHY		Supplier#000008423	aaJgDHqJ9,oq
Z		XivWaZvNeeD,0KdMrhA	
Supplier#000006872		Supplier#000008480	
4inbIrTbf3p4gU8NmOFN		Xz8nqXqo9MHFdHmFV7UP	
Supplier#000006949	Ffu26iJzkOgygMr1kII	Supplier#000008532	7OYRAX0Vu5OnclSU61
exZSXrw7		uK Wu49,IJm73xJ	
Supplier#000006985	c	Supplier#000008595	
66IQCaBt1cyKh0,1Wpvha6tAvROXj		fj,IpUXkaXtr64XdrnPoQAEO	
Supplier#000007072		Supplier#000008610	
Zy9t3SeZQrX9OEVUzTTRmZqdkSHFBg		9K5KbS,wbWWYz6d8KsfRtgv3j4qs5Uz5	
Supplier#000007098		Supplier#000008705	Rm0y
IXHSK0hoWcPPqxYd5CbjA3a4ep6NHATvKojdm		adNbu1WtID8nRcXoMPniC	
ux		Supplier#000008742	
Supplier#000007135	GzrnCh5T5VyFLatS5	kEbFansgobnO76f,W0GgB	
Supplier#000007160		Supplier#000008841	jrSVfyZzMGQKYu 9isE,
8Ankp7fpXO8Ai7UmgnwESp8WMXw0sv3IabP		Supplier#000008895	ZOjWCHLQf8277KS
Supplier#000007169		g9PPiauGENlxYm H	
zmORVoYECdS8S8WDOVVc0OFD4		Supplier#000008967	
Supplier#000007322	w	ZGvmjuekrTmvCsdjEq6mVEj,J3yA2OyFhe	
TNKzMVArfqHI20yvEIYetnG9e3Z		Supplier#000008972	
Supplier#000007365	WZuJ9dfwaei,VnDOy14y	wYwlUsnV21dXwIzc3zA5Mfqh7h	
Supplier#000007398	6SMmUD1,,cmd60	Supplier#000009032	Bg0y qU8NtXnsZpa6ldt
Supplier#000007402	X65wVTM tZAHEA8aV	Supplier#000009147	Acqx ujuB1SMcwu
Supplier#000007448	uJJB4JhITmiUaV5pQa	Supplier#000009252	
Supplier#000007477		mTO1Qc6jvJ,r1gtIXuEq64oZ3ObrxM,BVE0J	
SERH,wLJ4spw5juH60bBruv8j0K		Supplier#000009278	
Supplier#000007509		aA27sLuHRXpf3r,FO2LondcMLo	
BS05Ugh9CjiHjOcy8kTQg7eK			

Supplier#000009327
 yCAeNMTMTkIRNPaWX
 Supplier#000009430 JK9AEEMlYr
 Supplier#000009567
 AW4LuXOXuznqHOkITZjj2ptC EFrnRx sy3GwW
 Supplier#000009601
 WZEUXUPc09wVnDj5l6wfRO9uR
 Supplier#000009709
 A9DoPk2KnKGRb12Et4g53864,xgK
 Supplier#000009753
 wfJ5mP9ENTcGhlWmpDkgU1
 Supplier#000009796 t0tWiOE ZVocwb6B2k
 Supplier#000009799 sWvdH4kQWch4F
 Supplier#000009811
 nXlxtBT6D1v6TCb2iMYkyU
 Supplier#000009812
 rbI9euXFoPLIKQVYDVyRouslbbbKDHAKyXY
 Supplier#000009862 AhsvM5MdVS
 Supplier#000009868
 acHkKvVKwyHED66DMttTx
 Supplier#000009869 yOfAut
 Bp7aeMvk9eYRy0Ad7eY,KZ7yX3KKYEgQK
 Supplier#000009899 U3NBqk s Zz06al2m
 Supplier#000009974
 Uvh0hWngOu96WgB,OafBQOqwpWqzWG8

Number of rows retrieved is: 204

Stop timestamp 07/16/03 21:52:59.840000
 Query Time = 2.1 secs

Query 21

Start timestamp 07/16/03 21:53:32.110000

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

Tag: Q21 Stream: -1 Sequence number: 9

```

select
s_name,
count(*) as numwait
from
tpcd.supplier,
tpcd.lineitem l1,
tpcd.orders,
tpcd.nation
where
s_suppkey = l1.l_suppkey
  
```

```

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

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

Supplier#000001336 15
 Supplier#000001435 15
 Supplier#000003075 15
 Supplier#000003335 15
 Supplier#000005649 15
 Supplier#000006027 15
 Supplier#000006795 15
 Supplier#000006800 15
 Supplier#000006824 15
 Supplier#000007131 15
 Supplier#000007382 15
 Supplier#000008913 15
 Supplier#000009787 15
 Supplier#000000633 14
 Supplier#000001960 14
 Supplier#000002323 14
 Supplier#000002490 14
 Supplier#000002993 14
 Supplier#000003101 14
 Supplier#000004489 14
 Supplier#000005435 14
 Supplier#000005583 14
 Supplier#000005774 14
 Supplier#000007579 14
 Supplier#000008180 14
 Supplier#000008695 14
 Supplier#000009224 14
 Supplier#000000357 13
 Supplier#000000436 13
 Supplier#000000610 13
 Supplier#000000788 13
 Supplier#000000889 13
 Supplier#000001062 13
 Supplier#000001498 13
 Supplier#000002056 13
 Supplier#000002312 13
 Supplier#000002344 13
 Supplier#000002596 13
 Supplier#000002615 13
 Supplier#000002978 13
 Supplier#000003048 13
 Supplier#000003234 13
 Supplier#000003727 13
 Supplier#000003806 13
 Supplier#000004472 13
 Supplier#000005236 13
 Supplier#000005906 13
 Supplier#000006241 13
 Supplier#000006326 13
 Supplier#000006384 13
 Supplier#000006394 13
 Supplier#000006624 13
 Supplier#000006629 13
 Supplier#000006682 13
 Supplier#000006737 13
 Supplier#000006825 13

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

Number of rows retrieved is: 100

 Stop timestamp 07/16/03 21:53:36.110000
 Query Time = 4.0 secs

Query 22

Start timestamp 07/16/03 21:53:39.870000

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

Tag: Q22 Stream: -1 Sequence number: 12

```
select
  cntrycode,
  count(*) as numcust,
  sum(c_acctbal) as totacctbal
from
  (
  select
    substr(c_phone, 1, 2) as cntrycode,
    c_acctbal
  from
    tpcd.customer
  where
    substr(c_phone, 1, 2) in
```

```

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

```

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

Number of rows retrieved is: 7

Stop timestamp 07/16/03 21:53:41.440000
Query Time = 1.6 secs

First 10 Rows of the Database

```

SELECT * FROM TPCD.REGION FETCH FIRST
10 ROWS ONLY

```

```

R_REGIONKEY R_NAME
R_COMMENT

```

```

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

```

5 record(s) selected.

```

SELECT * FROM TPCD.NATION FETCH FIRST
10 ROWS ONLY

```

```

N_NATIONKEY N_NAME
N_REGIONKEY N_COMMENT

```

```

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

```

10 record(s) selected.

```

SELECT * FROM TPCD.PART FETCH FIRST 10
ROWS ONLY

```

P_PARTKEY P_NAME
P_MFGR P_BRAND P_TYPE
P_SIZE P_CONTAINER P_RETAILPRICE
P_COMMENT

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

10 record(s) selected.

SELECT * FROM TPCD.SUPPLIER FETCH
FIRST 10 ROWS ONLY

S_SUPPKEY S_NAME S_ADDRESS
S_NATIONKEY S_PHONE S_ACCTBAL
S_COMMENT

6 Supplier#000000006 tQxuVm7s7CnK
14 24-696-997-4969 +1.36579000000000E+003
even requests wake carefully! fluffily final pinto
beans run slyly among t
131 Supplier#000000131
u3mTHMgBC0yJTLufr01TuHImgflQUXv
14 24-293-181-3975 +1.30120000000000E+003
sometimes final accounts about the carefully
pending requests haggle blithely about the
216 Supplier#000000216 K83M7iWDJx N
Y 14 24-182-902-2539
+6.90254000000000E+003 foxes cajole pains. even
requests use quickly pendin
307 Supplier#000000307
3wL9YHFIVddxzh3mwy6SSRpfmzKvwAGmXK
14 24-499-938-5607 +2.16865000000000E+003
enticingly dogged packages wake. blit
492 Supplier#000000492 8wEulEYM
zGvMXfDNNEw4B 14 24-875-
296-5180 +8.36806000000000E+003 slyly express
accounts nag carefully. requests haggle slyly special
Tire
508 Supplier#000000508
F9,suuHYbe6kCRCPZaeSHSPAFBk9vOcFX8TUx
14 24-179-400-2422 +3.87822000000000E+003
deposits believe about the slyly final accounts.
furiously ironic theodo
588 Supplier#000000588 e3yF5zmSj y81I
14 24-180-601-5741 +9.76006000000000E+003
quickly pending packages haggle fu
717 Supplier#000000717
hhUrgvyxsdTfzGY4OrQSHeZmMNB2L75xk
14 24-797-880-9149 +6.74118000000000E+003
pending theodolites x-ray slowly among th
807 Supplier#000000807
CIHvM1nuPUESGg35Ls 14 24-
255-894-5069 +1.07797000000000E+003 ideas
sleep fluffily accor
1044 Supplier#000001044
imHHzVmeNI,OwowfxLg5lJzQOROqT
14 24-230-793-4577 +3.65482000000000E+003
pending, regular ideas doubt. fluffily ironic pinto
beans cajole blithely

10 record(s) selected.

SELECT * FROM TPCD.PARTSUPP FETCH
FIRST 10 ROWS ONLY

PS_PARTKEY PS_SUPPKEY PS_AVAILQTY
PS_SUPPLYCOST PS_COMMENT

--

9 10 7054
+8.4200000000000E+001 final ideas through the
requests boost quickly about the furiously regular
accounts. blithely silent foxes affix carefully ironic
instructions. blithely bold foxe

9 750010 7542
+8.1184000000000E+002 carefully unusual
dependencies cajole fluffily beyond the quickly bold

9 1500010 9583
+3.8131000000000E+002 carefully ironic pinto
beans nag against the quickly regular somas. regular,
ironic deposits wake

9 2250010 3063
+2.9184000000000E+002 doggedly ironic waters
are furiously. bold dependencie

12 13 3610
+6.5973000000000E+002 unusual dolphins sleep
slyly. ironic deposits use fluffily. carefully unusual
platelets poach slyly. evenly pending deposits nag
ironi

12 750013 7606
+3.3281000000000E+002 slyly daring foxes nag
slyly-- carefully special foxes integrate final
accounts. accounts sleep slyly furiously special
accounts. ironic,

12 1500013 824
+3.3706000000000E+002 quickly special pinto
beans haggle quickly above the e

12 2250013 5454
+9.0170000000000E+002 regular ideas sleep about
the ironic pinto beans. furious

19 20 1416
+1.4480000000000E+002 silent packages impress
fluffily slyly even theodolites. ironic deposits nag
carefully excuses. express deposits wake slyly.

19 750020 5467
+4.0570000000000E+002 regular packages cajole
slyly unusual packages. even deposits detect regular
deposits. ironic, special theodolit

10 record(s) selected.

SELECT * FROM TPCD.CUSTOMER FETCH
FIRST 10 ROWS ONLY

C_CUSTKEY C_NAME
C_ADDRESS C_NATIONKEY
C_PHONE C_ACCTBAL
C_MKTSEGMENT C_COMMENT

11 Customer#000000011 PkWS
3HIXqwTuzrKg633BEi 23 33-464-
151-3439 -2.7260000000000E+002 BUILDING
furiously express packages are. regular courts play
deposits. silent, ironic packages engage. furiously
regula
r a

87 Customer#000000087 hgGhHVSWQI
6jZ6Ev 23 33-869-884-7053
+6.3275400000000E+003 FURNITURE bold,
unusual excuses haggle daringly according to the
carefully express requests! theod

338 Customer#000000338
aiYAeWgI0okGSJv7OgvKqMvPLhxF3blT8josX
23 33-302-620-7535 +4.0924900000000E+003
FURNITURE carefully final excuses are
sometimes regular instructions.

364 Customer#000000364
SQ3b5Q5OtrmmZjJ87tq,o1TiXKVJQ0M7ZOuud
23 33-492-647-4972 +3.2240000000000E+001
HOUSEHOLD quickly quiet packages according
to the fluffily ironic deposits cajo

432 Customer#000000432 FDConiq
g20GI9dH QTM ZNX4OB9KU 23 33-
307-912-9016 +5.7156400000000E+003
BUILDING express, final dolphins wake regular
deposits. e

830 Customer#000000830
4fNmWCmfysljUI 23 33-408-

548-6806 +7.77565000000000E+003
AUTOMOBILE silent, ironic requests sleep along
the blithely regular instructions. final requests af

1067 Customer#000001067 g25CH,fhra
23 33-764-123-9568 +9.15384000000000E+003
FURNITURE final, bold waters after the carefully
silent ideas sleep even requests; fluffily unusual
requests across the
care

12874 Customer#000012874 LK
o1OAQpsLCI85vQSAg 23 33-
592-120-9012 +3.52163000000000E+003
HOUSEHOLD slyly even pinto beans mainta

13037 Customer#000013037
AjrKJRNKGDMfekT7 23 33-
821-137-5031 +2.53186000000000E+003
FURNITURE finally special foxes hang blithely
deposits. final deposits nag blithely? ironic, ironic
excuses sleep carefu
l

13076 Customer#000013076
sBwsw2,aQuTuGnt 23 33-995-
343-7109 +1.71650000000000E+002 BUILDING
slyly pending asymptotes above the slyly special
dugouts believe around the slyly special dugouts.
furious

10 record(s) selected.

SELECT * FROM TPCD.ORDERS FETCH FIRST
10 ROWS ONLY

O_ORDERKEY O_CUSTKEY
O_ORDERSTATUS O_TOTALPRICE
O_ORDERDATE O_ORDERPRIORITY
O_CLERK O_SHIPPRIORITY
O_COMMENT

59718 7424651 F
+1.12649120000000E+005 01/01/1992 5-LOW
Clerk#000269045 0 carefully regular pinto
beans across the even grouches dete
334181 8600005 F
+9.40402300000000E+004 01/01/1992 2-HIGH
Clerk#000061971 0 fluffily pending foxes
haggle carefully furiously final pinto beans. s
360261 28122308 F
+1.49784400000000E+004 01/01/1992 5-LOW

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

10 record(s) selected.

SELECT * FROM TPCD.LINEITEM FETCH
FIRST 10 ROWS ONLY

L_ORDERKEY L_PARTKEY L_SUPPKEY
L_LINENUMBER L_QUANTITY
L_EXTENDEDPRICE L_DISCOUNT
L_TAX L_RETURNFLAG
L_LINESTATUS L_SHIPDATE
L_COMMITDATE L_RECEIPTDATE
L_SHIPINSTRUCT L_SHIPMODE L_COM
MENT

1054181 4865078 1865079 1
+4.50000000000000E+001
+4.69273500000000E+004 +3.00000000000000E-
002 +8.00000000000000E-002 R F

01/02/1992 02/05/1992 01/15/1992 NONE
MAIL even
instructions kindle furio
5018977 24611600 611601 1
+2.000000000000000E+001
+3.020740000000000E+004
+0.000000000000000E+000
+0.000000000000000E+000 A F
01/02/1992 03/19/1992 01/15/1992 NONE
SHIP quick
ly ironic excu
14168833 46737982 988028 3
+4.900000000000000E+001
+9.886485000000000E+004 +9.000000000000000E-
002 +5.000000000000000E-002 R F
01/02/1992 02/01/1992 01/28/1992 TAKE
BACK RETURN SHIP furio
usly bold courts are careful
15413986 53978084 2978085 4
+1.400000000000000E+001
+1.623146000000000E+004 +5.000000000000000E-
002 +5.000000000000000E-002 A F
01/02/1992 01/31/1992 01/04/1992 COLLECT
COD TRUCK final
, bold dolphins
18436929 39951321 2451348 1
+5.000000000000000E+000
+6.851650000000000E+003
+0.000000000000000E+000 +7.000000000000000E-
002 A F 01/02/1992 02/27/1992
01/26/1992 TAKE BACK RETURN SHIP
even,
regular platelets na
19547653 37197566 1947579 1
+4.800000000000000E+001
+7.976208000000000E+004
+0.000000000000000E+000 +7.000000000000000E-
002 R F 01/02/1992 02/08/1992
01/26/1992 NONE MAIL regul
ar accounts hagggle. h
19918976 6334057 2584064 6
+3.100000000000000E+001
+3.381294000000000E+004 +5.000000000000000E-
002 +6.000000000000000E-002 R F
01/02/1992 02/23/1992 01/03/1992 DELIVER
IN PERSON REG AIR caref
ully fluffy requests
22142214 56458378 1708433 7
+1.900000000000000E+001
+2.533745000000000E+004
+0.000000000000000E+000 +7.000000000000000E-
002 R F 01/02/1992 03/17/1992
01/03/1992 NONE AIR caref
ully express requests sleep
22518080 50882189 1382222 6
+3.900000000000000E+001

+4.557696000000000E+004 +5.000000000000000E-
002 +7.000000000000000E-002 R F
01/02/1992 03/06/1992 01/30/1992 COLLECT
COD SHIP caref
ully special ideas
26601603 39300513 2550553 4
+2.000000000000000E+000
+3.023100000000000E+003
+0.000000000000000E+000 +2.000000000000000E-
002 R F 01/02/1992 03/28/1992
01/08/1992 COLLECT COD RAIL
speci
al platelets wake about the slyly fin

10 record(s) selected.

Query Substitution Parameters

Power stream Seed = 727170918
-- TPC TPC-H Parameter Substitution (Version
1.3.0)
-- using 727170918 as a seed to the RNG
Q1 DELTA 69
Q2 SIZE 40
TYPE TIN
REGION AFRICA
Q3 SEGMENT FURNITURE
DATE 1995-03-04
Q4 DATE 1993-08-01
Q5 REGION ASIA
DATE 1996-01-01
Q6 DATE 1996-01-01
DISCOUNT 0.04
QUANTITY 24
Q7 NATION1 INDONESIA
NATION2 ALGERIA
Q8 NATION ALGERIA
REGION AFRICA
TYPE SMALL ANODIZED STEEL
Q9 COLOR cornflower
Q10 DATE 1994-07-01
Q11 NATION BRAZIL
FRACTION 0.0000003333
Q12 SHIPMODE1 TRUCK
SHIPMODE2 RAIL
DATE 1994-01-01
Q13 WORD1 pending
WORD2 packages
Q14 DATE 1994-03-01
Q15 DATE 1997-03-01
Q16 BRAND Brand#41
TYPE ECONOMY PLATED
SIZE1 2

SIZE2 38
 SIZE3 36
 SIZE4 4
 SIZE5 26
 SIZE6 33
 SIZE7 46
 SIZE8 30
 Q17 BRAND Brand#14
 CONTAINER LG CASE
 Q18 QUANTITY 314
 Q19 BRAND1 Brand#22
 BRAND2 Brand#41
 BRAND3 Brand#13
 QUANTITY1 10
 QUANTITY2 10
 QUANTITY3 21
 Q20 COLOUR floral
 DATE 1996-01-01
 NATION ARGENTINA
 Q21 NATION PERU
 Q22 I1 30
 I2 11
 I3 13
 I4 32
 I5 24
 I6 33
 I7 17

 Throughput Stream = 1 Seed = 727170919
 -- TPC TPC-H Parameter Substitution (Version 1.3.0)
 -- using 727170919 as a seed to the RNG
 Q1 DELTA 77
 Q2 SIZE 28
 TYPE COPPER
 REGION EUROPE
 Q3 SEGMENT MACHINERY
 DATE 1995-03-20
 Q4 DATE 1996-03-01
 Q5 REGION EUROPE
 DATE 1997-01-01
 Q6 DATE 1997-01-01
 DISCOUNT 0.02
 QUANTITY 25
 Q7 NATION1 ARGENTINA
 NATION2 PERU
 Q8 NATION PERU
 REGION AMERICA
 TYPE STANDARD POLISHED STEEL
 Q9 COLOR burlywood
 Q10 DATE 1993-05-01
 Q11 NATION MOROCCO
 FRACTION 0.0000003333
 Q12 SHIPMODE1 RAIL
 SHIPMODE2 REG AIR
 DATE 1993-01-01

Q13 WORD1 pending
 WORD2 requests
 Q14 DATE 1994-07-01
 Q15 DATE 1994-12-01
 Q16 BRAND Brand#31
 TYPE STANDARD POLISHED
 SIZE1 12
 SIZE2 25
 SIZE3 1
 SIZE4 33
 SIZE5 48
 SIZE6 3
 SIZE7 8
 SIZE8 7
 Q17 BRAND Brand#11
 CONTAINER LG JAR
 Q18 QUANTITY 312
 Q19 BRAND1 Brand#24
 BRAND2 Brand#24
 BRAND3 Brand#52
 QUANTITY1 5
 QUANTITY2 11
 QUANTITY3 28
 Q20 COLOUR plum
 DATE 1994-01-01
 NATION MOZAMBIQUE
 Q21 NATION INDONESIA
 Q22 I1 26
 I2 13
 I3 19
 I4 14
 I5 21
 I6 16
 I7 25

 Throughput Stream = 2 Seed = 727170920
 -- TPC TPC-H Parameter Substitution (Version 1.3.0)
 -- using 727170920 as a seed to the RNG
 Q1 DELTA 85
 Q2 SIZE 15
 TYPE STEEL
 REGION AFRICA
 Q3 SEGMENT BUILDING
 DATE 1995-03-06
 Q4 DATE 1993-12-01
 Q5 REGION MIDDLE EAST
 DATE 1997-01-01
 Q6 DATE 1997-01-01
 DISCOUNT 0.07
 QUANTITY 24
 Q7 NATION1 CHINA
 NATION2 INDONESIA
 Q8 NATION INDONESIA
 REGION ASIA
 TYPE STANDARD BURNISHED STEEL

Q9 COLOR bisque
 Q10 DATE 1994-02-01
 Q11 NATION CANADA
 FRACTION 0.0000003333
 Q12 SHIPMODE1 AIR
 SHIPMODE2 RAIL
 DATE 1994-01-01
 Q13 WORD1 pending
 WORD2 requests
 Q14 DATE 1994-10-01
 Q15 DATE 1997-06-01
 Q16 BRAND Brand#11
 TYPE LARGE ANODIZED
 SIZE1 6
 SIZE2 7
 SIZE3 10
 SIZE4 8
 SIZE5 44
 SIZE6 37
 SIZE7 49
 SIZE8 11
 Q17 BRAND Brand#13
 CONTAINER LG CAN
 Q18 QUANTITY 314
 Q19 BRAND1 Brand#21
 BRAND2 Brand#12
 BRAND3 Brand#52
 QUANTITY1 10
 QUANTITY2 12
 QUANTITY3 24
 Q20 COLOUR burnished
 DATE 1993-01-01
 NATION FRANCE
 Q21 NATION ARGENTINA
 Q22 I1 16
 I2 25
 I3 21
 I4 34
 I5 23
 I6 22
 I7 31

Throughput Stream = 3 Seed = 727170921
 -- TPC TPC-H Parameter Substitution (Version
 1.3.0)
 -- using 727170921 as a seed to the RNG

Q1 DELTA 93
 Q2 SIZE 3
 TYPE BRASS
 REGION EUROPE
 Q3 SEGMENT MACHINERY
 DATE 1995-03-22
 Q4 DATE 1996-07-01
 Q5 REGION AFRICA
 DATE 1997-01-01
 Q6 DATE 1997-01-01

DISCOUNT 0.05
 QUANTITY 24
 Q7 NATION1 INDONESIA
 NATION2 ARGENTINA
 Q8 NATION ARGENTINA
 REGION AMERICA
 TYPE PROMO BRUSHED COPPER
 Q9 COLOR yellow
 Q10 DATE 1994-11-01
 Q11 NATION MOZAMBIQUE
 FRACTION 0.0000003333
 Q12 SHIPMODE1 REG AIR
 SHIPMODE2 TRUCK
 DATE 1994-01-01
 Q13 WORD1 pending
 WORD2 requests
 Q14 DATE 1995-01-01
 Q15 DATE 1995-03-01
 Q16 BRAND Brand#41
 TYPE PROMO BURNISHED
 SIZE1 21
 SIZE2 5
 SIZE3 17
 SIZE4 11
 SIZE5 6
 SIZE6 41
 SIZE7 14
 SIZE8 15
 Q17 BRAND Brand#15
 CONTAINER MED CASE
 Q18 QUANTITY 315
 Q19 BRAND1 Brand#33
 BRAND2 Brand#45
 BRAND3 Brand#51
 QUANTITY1 5
 QUANTITY2 13
 QUANTITY3 20
 Q20 COLOUR metallic
 DATE 1996-01-01
 NATION SAUDI ARABIA
 Q21 NATION ROMANIA
 Q22 I1 30
 I2 32
 I3 12
 I4 16
 I5 17
 I6 11
 I7 21

Throughput Stream = 4 Seed = 727170922
 -- TPC TPC-H Parameter Substitution (Version
 1.3.0)
 -- using 727170922 as a seed to the RNG
 Q1 DELTA 101
 Q2 SIZE 41
 TYPE TIN

REGION AMERICA
 Q3 SEGMENT BUILDING
 DATE 1995-03-08
 Q4 DATE 1994-03-01
 Q5 REGION ASIA
 DATE 1997-01-01
 Q6 DATE 1997-01-01
 DISCOUNT 0.02
 QUANTITY 25
 Q7 NATION1 ARGENTINA
 NATION2 CHINA
 Q8 NATION CHINA
 REGION ASIA
 TYPE PROMO PLATED COPPER
 Q9 COLOR thistle
 Q10 DATE 1993-08-01
 Q11 NATION EGYPT
 FRACTION 0.0000003333
 Q12 SHIPMODE1 SHIP
 SHIPMODE2 TRUCK
 DATE 1995-01-01
 Q13 WORD1 pending
 WORD2 requests
 Q14 DATE 1995-04-01
 Q15 DATE 1997-10-01
 Q16 BRAND Brand#31
 TYPE SMALL POLISHED
 SIZE1 25
 SIZE2 33
 SIZE3 23
 SIZE4 11
 SIZE5 37
 SIZE6 1
 SIZE7 49
 SIZE8 21
 Q17 BRAND Brand#12
 CONTAINER MED JAR
 Q18 QUANTITY 313
 Q19 BRAND1 Brand#35
 BRAND2 Brand#33
 BRAND3 Brand#45
 QUANTITY1 1
 QUANTITY2 14
 QUANTITY3 28
 Q20 COLOUR wheat
 DATE 1995-01-01
 NATION IRAN
 Q21 NATION IRAQ
 Q22 I1 24
 I2 17
 I3 23
 I4 20
 I5 21
 I6 15
 I7 19

Throughput Stream = 5 Seed = 727170923
 -- TPC TPC-H Parameter Substitution (Version 1.3.0)
 -- using 727170923 as a seed to the RNG
 Q1 DELTA 109
 Q2 SIZE 29
 TYPE COPPER
 REGION EUROPE
 Q3 SEGMENT HOUSEHOLD
 DATE 1995-03-24
 Q4 DATE 1996-10-01
 Q5 REGION EUROPE
 DATE 1997-01-01
 Q6 DATE 1997-01-01
 DISCOUNT 0.08
 QUANTITY 24
 Q7 NATION1 CHINA
 NATION2 IRAN
 Q8 NATION IRAN
 REGION MIDDLE EAST
 TYPE PROMO ANODIZED COPPER
 Q9 COLOR slate
 Q10 DATE 1994-05-01
 Q11 NATION PERU
 FRACTION 0.0000003333
 Q12 SHIPMODE1 MAIL
 SHIPMODE2 TRUCK
 DATE 1995-01-01
 Q13 WORD1 unusual
 WORD2 accounts
 Q14 DATE 1995-08-01
 Q15 DATE 1995-07-01
 Q16 BRAND Brand#11
 TYPE LARGE BRUSHED
 SIZE1 4
 SIZE2 13
 SIZE3 43
 SIZE4 28
 SIZE5 34
 SIZE6 32
 SIZE7 1
 SIZE8 42
 Q17 BRAND Brand#14
 CONTAINER MED CAN
 Q18 QUANTITY 314
 Q19 BRAND1 Brand#32
 BRAND2 Brand#11
 BRAND3 Brand#44
 QUANTITY1 6
 QUANTITY2 15
 QUANTITY3 24
 Q20 COLOUR honeydew
 DATE 1993-01-01
 NATION ALGERIA
 Q21 NATION CANADA
 Q22 I1 15

I2 26
 I3 29
 I4 23
 I5 18
 I6 25
 I7 21

 Throughput Stream = 6 Seed = 727170924
 -- TPC TPC-H Parameter Substitution (Version 1.3.0)
 -- using 727170924 as a seed to the RNG
 Q1 DELTA 117
 Q2 SIZE 16
 TYPE STEEL
 REGION AMERICA
 Q3 SEGMENT BUILDING
 DATE 1995-03-10
 Q4 DATE 1994-07-01
 Q5 REGION MIDDLE EAST
 DATE 1993-01-01
 Q6 DATE 1993-01-01
 DISCOUNT 0.05
 QUANTITY 24
 Q7 NATION1 IRAN
 NATION2 BRAZIL
 Q8 NATION BRAZIL
 REGION AMERICA
 TYPE ECONOMY POLISHED COPPER
 Q9 COLOR saddle
 Q10 DATE 1993-03-01
 Q11 NATION ETHIOPIA
 FRACTION 0.0000003333
 Q12 SHIPMODE1 TRUCK
 SHIPMODE2 REG AIR
 DATE 1996-01-01
 Q13 WORD1 unusual
 WORD2 accounts
 Q14 DATE 1995-11-01
 Q15 DATE 1993-03-01
 Q16 BRAND Brand#41
 TYPE STANDARD BURNISHED
 SIZE1 22
 SIZE2 2
 SIZE3 3
 SIZE4 35
 SIZE5 4
 SIZE6 32
 SIZE7 1
 SIZE8 29
 Q17 BRAND Brand#11
 CONTAINER JUMBO CASE
 Q18 QUANTITY 312
 Q19 BRAND1 Brand#45
 BRAND2 Brand#54
 BRAND3 Brand#34
 QUANTITY1 1

QUANTITY2 16
 QUANTITY3 20
 Q20 COLOUR saddle
 DATE 1997-01-01
 NATION MOROCCO
 Q21 NATION SAUDI ARABIA
 Q22 I1 20
 I2 13
 I3 10
 I4 22
 I5 11
 I6 25
 I7 27

Appendix D: Driver Source Code

ploaduf1

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

ploaduf2

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

load_line_uf

```
#!/bin/ksh
RFpair=$1;
db2 connect to tpcd
db2 "load from lineitem.tbl.u$RFpair of del
modified by coldel| fastparse messages /dev/null
replace into TPCDTEMP.LINEITEM_new
nonrecoverable partitioned db config mode
load_only part_file_location /home/ufdata;"
db2 commit;
db2 connect reset
db2 terminate
```

load_orders_uf

```
#!/bin/ksh
RFpair=$1;
db2 connect to tpcd
db2 "load from orders.tbl.u$RFpair of del modified
by coldel| fastparse messages /dev/null replace into
TPCDTEMP.ORDERS_new nonrecoverable
partitioned db config mode load_only
part_file_location /home/ufdata; "
db2 commit;
```

```
db2 connect reset
db2 terminate
```

preload_uf.ksh

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

```
toolsDir=/home/tpch/tpcd/tools
```

```
${toolsDir}/ploaduf1 21
${toolsDir}/ploaduf2 21
```

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

runpower

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

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

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

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



```

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

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

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

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

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

```

```

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

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

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

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

if ( $RealAudit ne "yes" )
{
    system("db2 activate database $dbname");
}

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

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

if ($platform eq "aix")
{
    # Create the sysunused file. This reports what
disks are attached, and which
# ones are being used. Its use spans both the
runpower and runthroughput tests
    system("echo \"The following disks are assigned to
the indicated volume groups\" >
$runDir/sysunused$runNum") && die "cannot
create $runDir/sysunused$runNum";

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

&getConfig("p");
if ( $rootPriv eq "yes" )
{
    # get the o/s tuning parameters...currently AIX
only and only if your
# user has root privileges to run this
    &getOSTune("p");
}
if ($gatherstats eq "on")
{
    # gather vm io and net stats
    if ($platform eq "aix" || $platform eq "sun" ||
$platform eq "ptx" ||
    $platform eq "hp" || $platform eq "linux")
    {
        # gather vmstats and iostats (and net stats if in
mpp mode)
        system("perl getstats p &");
    }
}
}
}

```

```

    }
    else
    {
        print "Stats gather not set up for current
platform $platform\n";
    }
}

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

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

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

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

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

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

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

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

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

if ( $product eq "pe" )

```

```

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

#####

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

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

# concatenate all the throughput inter files that were
used to
# generate these results into the calcmetrics output
file (mpinterX.metrics)
#cd $TPCD_RUN_DIR
&cat("$runDir${delim}mpqinter*","$runDir${delim}
mpinter$runNum.metrics");
if ( $runUF eq "UF" ) {
  &cat("$runDir${delim}mpufinter*","$runDir${delim}
mpinter$runNum.metrics");
}
}
# if ( $runUF eq "no" ) {
#   &rm("$runDir${delim}mpuf*");
# }

#####

# no longer activate/deactivate the database
# if ( $RealAudit ne "yes" )
#{
#   # deactivate the database
#   system("db2 deactivate database $dbname");
# }

# do not stop the database after the power test
# if ( $RealAudit ne "yes" )
#{
#   system("db2stop");
# }

1;

sub getConfig
{
  $stesttype=${_}[0];
  print "Getting database configuration.\n";

  $dbtunefile="$runDir${delim}m${testtype}dbtune$
{runNum}";
  open(DBTUNE, ">$dbtunefile") || die "Can't open
$dbtunefile: $!\n";
  $timestamp=`perl gettimestamp "long"`;
  print DBTUNE "Database and Database manager
configuration taken at : $timestamp";
  close(DBTUNE);
  system("db2level >> $dbtunefile");
  system("db2 get database configuration for
$dbname >> $dbtunefile");
  system("db2 get database manager configuration
>> $dbtunefile");
  system("db2set >> $dbtunefile");
  if (( $mode eq "mln" ) || ( $mode eq "mpp" ))
  {
    $cfgfile="$runDir${delim}dbtune${runNum}. ";
    #removed by Alex due to hang
    #system("db2_all \"\|\" typeset -i ln=##; db2 get db
cfg for $dbname > $cfgfile${ln} ; db2 get dbm cfg

```

```

>> $cfgfile\${ln}; db2set >> $cfgfile\${ln}; db2
terminate "");
}

}

sub getOSTune
{
    $stesttype=${_}[0];
    if ( $platform eq "aix" )
    {
        print "Getting OS and VMdatabase
configuration.\n";

        $ostunefile="$runDir${delim}m${testtype}ostune$
{runNum}";
        open(OSTUNE, ">$ostunefile") || die "Can't
open $ostunefile: $!\n";
        $timestamp=`perl gettimestamp "long"`;
        print OSTUNE "Operating System and Virtual
Memory configuration taken at : $timestamp";
        close(OSTUNE);

        system("$delim}usr${delim}samples${delim}kern
el${delim}schedtune >> $ostunefile");

        system("$delim}usr${delim}samples${delim}kern
el${delim}vmtune >> $ostunefile");
    }
    else
    {
        print "OS parameters retrieval not supported for
$platform \n";
    }
}

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

```

runthroughput

```

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

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

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

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

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

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

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

```

```

&setVar(@reqVars, "ERROR");

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

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

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

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

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

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

}

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

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

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

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

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

if ( $platform eq "aix" || $platform eq "sun" ||
$platform eq "nt" || $platform eq "hp" || $platform eq
"linux")
{

```

```

    system("$streamExNT perl watchstreams
$streamEx");
}
else
{
    die "platform not supported, can't start
watchstreams in background";
}

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

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

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

$loopStream=1;

for ( $loopStream = 1; $loopStream <=
$numStream; $loopStream++)
{
    print "starting stream $loopStream\n";
    system("echo Executing stream $loopStream out
of $numStream.");
    # run the queries
    if ( $platform eq "aix" || $platform eq "sun" ||
$platform eq "nt" || $platform eq "ptx" ||
    $platform eq "hp" || $platform eq "linux")
    {
        system("$streamExNT $path${delim}tpcdbatch
-d $dbname -f
$runDir${delim}qtextt$loopStream.sql -r on -b on -s
$sf -u t1 -m $inlistmax -n $loopStream $streamEx");
    }
}

```

```

}
else
{
    die "platform $platform not supported yet";
}

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

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

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

#if ($platform eq "aix")
#{
# show the disks that are used/unused
#   system("getdisks \"After the completion of the
Throughput Test\");
#}
if ($gatherstats eq "on")
{
    # gather vm io and net stats
    if ($platform eq "aix" || $platform eq "sun" ||
$platform eq "ptx" || $platform eq "linux")
    {
        # kill the stats that were being gathered
        if ($platform eq "ptx")
        {
            $src= `perl5 zap -f "sar"`;
            $src= `perl5 zap -f "sadc"`;
        }
    }
    else
    {

```

```

        $src= `perl5 zap "-f" "vmstat"`;
        $src= `perl5 zap "-f" "iostat"`;
    }
    if ( $pn > 1 )
    {
        $src= `perl5 zap "-f" "netstat"`;
    }
    $src= `perl5 zap "-f" "getstats"`;
}

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

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

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

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

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

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

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

if ($runUF ne "no") {

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

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

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

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

# deactivate the database this needs to remain at the
end of run throughput so
# asynchronous writing of the log files completes.
system("db2 deactivate database $dbname");
$src=&dodb_noconn("db2 get db cfg for $dbname |
grep -i log >>
$runDir${delim}endLog.Info", $all_In);
if ( $logDir ne "NULL" )
{
    $src=&dodb_noconn("$dircmd $logDir >>
$runDir${delim}endLog.Info", $all_In);
}

#system("db2_all \`}db2 get db cfg for tpcd | grep -i
log >> $runDir${delim}endLog.Info ; db2
terminate\` ");

```



```

#system("ls -ltra /node??vg.log/NODE00* >>
$runDir${delim}endLog.Info");

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

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

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

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

1;

sub getConfig
{
    $testtype=$_[0];
    print "Getting database configuration.\n";

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

sub getOSTune
{
    $testtype=$_[0];
    if ( $platform eq "aix" || $platform eq "linux" )
    {
        print "Getting OS and VMdatabase
configuration.\n";

```

```

$ostunefile="$runDir${delim}m${testtype}ostune$
{runNum}";
    open(OSTUNE, ">$ostunefile") || die "Can't
open $ostunefile: $!\n";
    $timestamp=`perl gettimestamp "long"`;
    print OSTUNE "Operating System and Virtual
Memory configuration taken at : $timestamp";
    close(OSTUNE);

system("${delim}usr${delim}samples${delim}kern
el${delim}schedtune >> $ostunefile");

system("${delim}usr${delim}samples${delim}kern
el${delim}vmtune >> $ostunefile");
}
else
{
    print "OS parameters retrieval not supported for
$platform \n";
}
}

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

```

tpcdbatch.h

```

/*****
*****
*
*   TPCDBATCH.H
*
*   Revision History:
*
*   27 may 99 bbe from (24 nov 98 jen)
fixNTimestamp - fixed NT timestamp to print
millisecond correctly

```

```

* 27 may 99 bbe from (10 dec 98 jen) SUN - added
Haider's changes necessary for SUN
* 17 jun 99 jen Increased version to 5.1
* 10 aug 99 bbe Increased version to 5.2
* 13 aug 99 bbe Increased version to 5.3
* 18 mar 02 ken Increased version to 5.7
*****
*****/

/** Necessary header files **/

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

#include <time.h>
#include <ctype.h>
#if (defined(SQLAIX) || defined(SQLPTX) ||
defined(LINUX) || defined(SQLHP))
#include <unistd.h> /* SUN */
#include <sys/stat.h> /* SUN */
#endif
#if ((defined(SQLAIX) || defined(SQLPTX)) &&
!defined(LINUX))
#include <sys/vnode.h> /* SUN */
#endif
#ifndef SQLWINT
#include <sys/time.h>
/*@d33143aha*/
#include <sys/ipc.h>
#include <sys/sem.h>
#if (!defined(SQLPTX) && !defined(LINUX)&&
!defined(SQLHP))
#include <sys/mode.h>
#endif
#include <sys/timeb.h>
#include <sys/types.h>
#else
#include <windows.h>
#include <sys\timeb.h>
#endif
#include <errno.h>

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

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

*****
*****/

/* Define synonyms here
*/
*****
*****/
#define TPCDBATCH_VERSION "5.7"

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

#define TPCDBATCH_MAX_COLS 100
/* @d30369 tlg */

#define TPCDBATCH_CHAR char

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

#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"
#endif
#define TPCD_NONPARTITIONED
#define UF2OUTSTREAMPATTERN
"%s%cuf2.%02d.%d.out"

```

```

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

#define T_STAMP_FORM_1 1
#define T_STAMP_FORM_2 2
/* jen TIME_ACC start */
#define T_STAMP_FORM_3 3
#define T_STAMP_1LEN 17
#if defined (SQLUNIX) || defined (SQLAIX) ||
defined (SQLHP)
#define T_STAMP_3LEN 24
#elif defined (SQLOS2) || defined (SQLWINT) ||
defined (SQLWIN) || defined (SQLDOS))
#define T_STAMP_3LEN 21 /* WIN NT
timestamp fix bbe */
#else
#error Unknown operating system
#endif
/* jen TIME_ACC start */

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

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

/** Defines needed for decimal conversion */
#define SQLZ_DYNLINK
#define TRUE 1
#define LEFT 1
#define RIGHT 0
#define FALSE 0

```

```

#define sqlrx_get_left_nibble(byte) (((unsigned
char)(byte)) >> 4)

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

/** Timer-necessary defines for portability */
#if defined (SQLOS2) || defined (SQLWINT) ||
defined (SQLWIN) || defined (SQLDOS)
typedef struct timeb Timer_struct;
#elif defined (SQLUNIX) || defined (SQLAIX) ||
defined (SQLHP) /*TIMER jen*/
typedef struct timeval Timer_struct;
#else
#error Unknown operating system
#endif

/* sleep time between starting subsequent
tpcdbatches running UF1 and UF2 */
#define UF1_SLEEP 1
#define UF2_SLEEP 1
#define UF_DEADLOCK_SLEEP 1 /* sleep
between deadlock retries in UF1,UF2 */

#define MAXWAIT 50 /* maximum retries for
deadlock encounters */

#define DEBUG 0 /* to be set to 1 for diagnostic
purposes if needed */
/* #define UF1DEBUG 1 */
/* #define UF2DEBUG 1 */

```

tpcdbatch.sqc

```

/*****
*****
*
* TPCDBATCH.SQC
*
* Revision History:
*
* 21 Dec 95 jen Corrected calculation of geometric
mean to include in the
* count of statements the update functions.
* 03 Jan 96 jen Corrected calculation of arithmetic
mean to not include the
* timings for the update functions. (only
want query timings
* as part of arithmetic mean)
* 15 Jan 96 jen Added extra timestamps to the
update functions.

```

* 22 Jan 96 jen Get rid of checking of short_time....we always use the long

* timings.

* Fixed timings to print query/uf times rounded up to 0.1 seconds

* and uses these rounded time values in subsequent calculations

* Fixed bug where last seed in mseedme file wasn't getting read

* correctly - EOF processing done too soon.

* 22 Feb 96 kbs port to NT

* 26 Mar 96 kbs Fix to avoid counting UFs as queries for min max

* 27 Jun 97 wlc Temporarily fixed deadlock problems when doing UF1, UF2

* 30 Jul 97 wlc Add in support for load_update and TPCD_SPLIT_DELETES

* 13 Aug 97 wlc fixed UF1 log file formatting problem,

* using TPCD_TMP_DIR for temp files instead of /tmp,

* make summary table fit in 80-column,

* fixed UF2 # of deleted rows reporting problem

* 18 Aug 97 wlc added command line support for inlistmax

* 20 Aug 97 wlc added support for runthroughput without UF

* 27 Aug 97 aph Replaced hardcoded 'tpcaudit' with getenv("TPCD_AUDIT_DIR")

* 05 Sep 97 wlc fixing free() problem in NT

* 26 Sep 97 kmw change FLOAT processing in echo_sqlda and print_headings

* 10 Oct 97 jen add lock table in share mode for staging tables

* 21 Oct 97 jen added explicit rollback on failure of uf1

* 27 Oct 97 jen don't update TPCD.xxxx.update.pair.num if not running UFs in

* throughput run

* 01 Nov 97 jen temp code to do a prep then execute stmt in UFs so we can

* get timings

* 03 Nov 97 jen realigned UF code for readability

* pushed UF2 commit into loop for inlistmax

* fixed UF2 code so rollback performed

* 04 Nov 97 jen Added code to handle vldb

* 06 Nov 97 jen Commented out temp code for prep then execute stmts using

* TPCD_PREPARETIME def

* Updated version number to 2.2

* send all output during update functions to output files, not

* stderr

* 10 Nov 97 jen jenCI Updated version number to 2.3

* Added handling of TPCD_CONCURRENT_INSERTS. Change control of

* chunk processing to use the concurrent_inserts value as the

* control. Now the inserts will be run in TPCD_CONCURRENT_INSERTS

* sets, each having concurrent_inserts/

* 13 Nov 97 jen jen DEADLOCK. Fixed bug that Alex found where deadlock count

* (maxwait) was incremented on every execution of the stmt as

* opposed to just when deadlock really happened.

* 14 Nov 97 jen jenSEM - fix up error reporting on semaphore failure

* sem_op now returns failure to caller so caller can report where

* failure has happened.

* Forced dbname to be upper case, an all other parts of update

* pair number to be lowercase

* 15 Nov 97 jen SEED Reworked code to grab the seed from the seed file. Now

* reusing seeds between runs, so power run will always use first

* seed, throughput will use the 2nd - #stream+1 seeds

* 13 Jan 98 jen LONG Increase stmt_str to be able to hold inlists with larger

* order key numbers

* 04 Mar 98 jen IMPORT added support for TPCD_UPDATE_IMPORT to chose whether

* using import or load api's for loading data into the staging

* tables

* 04 Mar 98 jen TIMER changed from using gettimeofday to gettimeofday for unix

* 01 Apr 98 jen Fixed IMPORT code to do the proper checking on strcmp (ie !strcmp)

* 01 Apr 98 jen removed code to handle vldb - not needed

* Upgraded version to 2.4 for (chunk

* 01 Apr 98 jen Fixed up import code on NT so the variable is recognized in the

* children

* 25 May 98 sks Reworked some of the environment variable code so consolidate as

* much as possible. Not all complete because of differences in

* the way nt and AIX calls (and starts stuff in background) for UFs

```

* 29 may 98 jen REUSE_STAGE Changed UF1 so
we reuse the same staging tables
*      instead of having a new set for each
update pair
* 06 jul 98 jen Removed locking of staging tables
since they are created with
*      locksize table now
* 06 jul 98 jen 912RETRY - added code to retry
query execution on 912 as well
*      as 911
* 07 jul 98 jen Fixed summary_table() so 1000x
adjustment not based on UF (setting
*      of max and min pointers
*      Added generic SleepSome function to
handle NT vs AIX sleep differences
* 01 apr 98 djd Added change to permit the use of
table functions for UF1.
*      to enable this set
TPCD_UPDATE_IMPORT to tf in TPCD.SETUP
file.
*      MERGED this into base copy on Jul 07
* 10 jul 98 jen haider's fix for 'outstream' var for
error processing in
*      runUF1_fn and runUF2_fn
*      Updated version to 2.5
* 25 sep 98 jen Added stream number printing into
mpqry* files and increases
*      accuracy of timestamp in mpqry (and
mts*qry*) files
* 06 oct 98 jen TIME_ACC Added accuracy of
timestamp in mpqry (and mts*qry*)
*      files. Cleaned up misuse of Sleep and
flushed buffers on
*      deadlocks
* 19 oct 98 kbs fix UF2_fn to correctly count rows
deleted in case of deadlock
* 20 oct 98 kbs rewrite UF2 and UF2_fn for static
SQL with staging table
* 23 oct 98 jen Cleaned up retrying of
order/lineitem on lineitem deadlock in UF1
* 24 oct 98 jen Used load_uf1 and load_uf2 instead
of general load_updates
* 26 oct 98 kbs inject the UF1 with a single staging
table
* 02 nov 98 jen Fixed processing of multiple
chunks in uf2 so don't duplicate
* 21 nov 98 kmw Fixed BIGINT
* 05 dec 98 aph Moved runUF1_fn() and
runUF2_fn() into a separate file tpcdUF.sqc
*      so that it can be bound separately with a
different isolation level.
* 21 dec 98 aph Integrated Jennifer's QppD
calculation (rounding & adjustment) fixes.
* 22 dec 98 aph For UFs during Throughput run,
defer CONNECT until children launched.

* 28 dec 98 aph Removed error_check() call after
CONNECT RESET
* 29 dec 98 aph For UFs do not COMMIT in
tpcdbatch.sqc. COMMITs happen in tpcdUF.sqc.
* 18 jan 99 kal replaced header with #include
"tpcdbatch.h"
* 27 may 99 bbeaton from (03 mar 99 jen) Fixed
SUN fix that wasn't compatible with
*      NT (using %D %T instead of %x %X for
strftime)
* 16 jun 99 jen Added missing LPCTSTR cast of
semaphore file name for NT
* 17 jun 99 jen SEMA Changes semaphore file for
update functions to look for tpcd.setup
*      not for the orders.*** update data file
* 21 jul 99 bbeaton Added semaphore control that
allows runpower to be run as two
*      separate streams (update and query).
This involves the use of
*      two semaphores to be used as it executes
in three different
*      sections. The first is the update inserts.
The next is the query
*      stream which is started with the update
stream, but waits until
*      the inserts are complete. The third
section is the update deletes
*      which execute after the queries are
complete.
* 21 jul 99 bbeaton Added functions to handle
semaphore creation, control, etc.
* 21 jul 99 bbeaton Modified output to mp*inter
files. It now only outputs
*      intermediate data that will be calculated
by calcmetric.pl. This
*      is a result of the runpower being split into
two streams and thus
*      tpcdbatch not having access to all data.
* 21 jul 99 bbeaton The start time for runpower
UF2 now does not start until after
*      the query stream is complete so that its
wait time is not included
*      NOTE: The wait time that the first UF1
in runthroughput still
*      includes the wait period that occurs
waiting on queries.
* 18 mar 02 kentond removed the need for list files.
Instead of using the *.list
*      files to determine the name of the output
files, the tags for the
*      source sql files are used.
*****
*****/
/* included in tpcdbatch.sqc and tpcdUF.sqc */

```

```

#include "tpcdbatch.h"

/*****
*****/
/* global structure containing elements passed
between different functions */
/*****
*****/
struct global_struct
{
    struct stmt_info  *s_info_ptr;      /* ptr to
stmt_info list */
    struct stmt_info  *s_info_stop_ptr; /* ptr to
last struct in list */
    struct comm_line_opt *c_l_opt;      /* ptr to
comm_line_opt struct */
    struct ctrl_flags  *c_flags;        /* ptr to
ctrl_flags struct */
    Timer_struct      stream_start_time; /* start
time for stream TIME_ACC */
    Timer_struct      stream_end_time;   /* end
time for stream TIME_ACC */
    char               file_time_stamp[50]; /* time
stamp for output files */
    double             scale_factor;     /* scale factor
of database */
    char               run_dir[150];     /* directory for
output files */
    int                copy_on_load;     /* indication of
whether or not */
                                /* to do use a copy
directory */
                                /* (equiv to COPY
YES) on load */
                                /* default is FALSE */
    long               lSeed;           /* seed used to
generate the */
                                /* queries for this
particular */
                                /* run.
*/
    FILE               *stream_list;    /* ptr to query
list file */
    char               update_num_file[150]; /* name of
file that keeps track */
                                /* of which update
pairs have run*/
    char               sem_file[150];   /* semaphore
name */
    char               sem_file2[150];  /* semaphore
name bbe */
    FILE               *stream_report_file; /* file to
report start stop */
                                /* progress of the
stream */
};

/*****
*****/
/* New type declaration to store details about SQL
statement */
/*****
*****/
struct stmt_info
{
    long               max_rows_fetch;
    long               max_rows_out;
    int                query_block;      /*
@d30369 tjj */
    unsigned int       stmt_num;        /*
@d24993 tjj */
    double             elapse_time;     /*
@d24993 tjj */
    double             adjusted_time;
    char               start_stamp[50]; /* start time
stamp for block */
    char               end_stamp[50];   /* end time
stamp for block */
    char               tag[50];         /* block tag
*/
    char               qry_description[100];
    struct stmt_info  *next;           /*
@d24993 tjj */
};

/*****
*****/
/* Structure containing command line options
*/
/*****
*****/
struct comm_line_opt
{
                                /* @d22275
tjj */
    /* 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;
};

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

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

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

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

void dumpCa(struct sqlca*); /*kmw*/

void display_usage(void);
char *uppercase(char *string);
char *lowercase(char *string);
void comm_line_parse(int agrc, char *argv[], struct
global_struct *g_struct);
int sqlrxd2a(char *decptr,char *asciiptr,short
prec,short scal);
void init_setup(int argc, char *argv[], struct
global_struct *g_struct);
void runUF1( struct global_struct *g_struct, int
updatePair );
void runUF2( struct global_struct *g_struct, int
updatePair );

/* These need to be extern because they're in another
SQC file. aph 981205 */
/*extern void runUF1_fn( int updatePair, int i );*/
/* aph 981205 */
/*extern void runUF2_fn( int updatePair, int i, int
numChunks );*/ /* aph 981205 */
/* Added four new arguments because SQL host
vars can't be global. aph 981205 */
extern void runUF1_fn ( int updatePair, int i, char
*dbname, char *userid, char *passwd );
extern void runUF2_fn ( int updatePair, int
thisConcurrentDelete, int numChunks, char
*dbname, char *userid, char *passwd );

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

char *get_time_stamp(int form, Timer_struct
*timer_pointer); /* TIME_ACC jen */
void summary_table (struct global_struct *g_struct);
void free_sqlda (struct sqlda *sqlda, int
select_status); /* @d30369 tjg */
void output_file(struct global_struct *g_struct);
int PreSQLprocess(struct global_struct *g_struct,
Timer_struct *start_time);
void SQLprocess(struct global_struct *g_struct);
int PostSQLprocess(struct global_struct *g_struct,
Timer_struct *start_time);
int cleanup(struct global_struct *g_struct);

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

```

```

HANDLE open_semaphore(struct global_struct
*g_struct, int num);
#else
int open_semaphore(struct global_struct *g_struct);
#endif

```

```
EXEC SQL INCLUDE SQLCA;
```

```

/*****
*****/
/* Declare the SQL host variables.
*/
/*****
*****/

```

```
EXEC SQL BEGIN DECLARE SECTION;
```

```

char stmt_str1[4000] = "\0"; /* Assume max
SQL statment

```

```

of 4000 char */
/* jen LONG */
struct {
short len;
char data[32700];
} stmt_str; /* jen LONG */

```

```

char dbname[9] = "\0";
char userid[9] = "\0";
char passwd[9] = "\0";
char sourcefile[256]; /* used for
semaphores and table functions?*/
sqlint32 chunk = 0; /* jenCI counter for
within the set of chunks*/

```

```
EXEC SQL END DECLARE SECTION;
```

```

/*****
*****/
/* Declare the global variables.
*/
/*****
*****/

```

```

struct sqllda *sqllda; /* SQL Descriptor
area */

```

```
/* Global environment variables (sks May 25 98)*/
```

```

char env_tpcd_dbname[100];
char env_user[100];
char env_tpcd_audit_dir[150];
char env_tpcd_path_delim[2];
char env_tpcd_tmp_dir[150];
char env_tpcd_run_on_multiple_nodes[10];
char env_tpcd_copy_dir[150];
char env_tpcd_update_import[10];

```

```
/* Other globals */
```

```

FILE *instream, *outstream; /* File
pointers
*/
int verbose = 0; /* Verbose option
flag */
int semcontrol = 1; /*
allows/disallows smaphores usage */
int updatePairStart; /* update pair to
start at */
int currentUpdatePair; /* update pair
running */
int updatePairStop; /* update pair to
stop before */
char newtime[50]="\0"; /* Des - moved
from get_time_stamp */
char ostreamfilename[256]; /* store
filename of ostream
wlc 081397 */
int inlistmax = 400; /* define # of keys
to delete at a time
wlc 081897 */
int sqlda_allocated = 0; /* fixing free()
problem in NT
wlc 090597 */
int iImportStagingTbl=0; /* IMPORT use
import or load (default) */
char temp_time_stamp[50]; /* holds end
timestamp to be copied into start_time_stamp of
next query bbeaton */
Timer_struct temp_time_struct; /* holds end
time value to be copied into start_time of next query
bbeaton */

```

```
/* constants for the semaphores used; 1 for
throughput and 2 for power */
```

```

#define INSERT_POWER_SEM 1
#define QUERY_POWER_SEM 2
#define THROUGHPUT_SEM 1

```

```

/*****
*****/
/* Start main program processing.
*/
/*****
*****/

```

```

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

```



```

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

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

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

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

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

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

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

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

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

strcpy(temp_time_stamp, "0");

```

```

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

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

/*****
*****

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

c_flags.eo_block = 0;

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

fprintf( ostream,"Start timestamp %*.*s \n",
T_STAMP_3LEN,T_STAMP_3LEN,
/* TIME_ACC jen*/
g_struct.s_info_ptr->start_stamp);
if (c_l_opt.intStreamNum >= 0)

```

```

{
  if (g_struct.lSeed == -1)
  {
    fprintf( ostream,"Using default qgen seed
file");
  }
  else
    fprintf( ostream,"Seed used =
%ld",g_struct.lSeed);

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

/** Read in the next statment **/

c_flags.select_status=Get_SQL_stmt(&g_struct);

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

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

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

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

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

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

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

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

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

return(0);

} /* end of main */

/*****
*****
/* Generic form of Sleep */
int SleepSome( int amount)
{
#ifdef SQLWINT
  sleep (amount);
#else
  Sleep (amount*1000); /* 10x for NT DJD
Changed "sleep" to "Sleep" */
#endif
  return 0;
}

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

```

```

/* Get environment variables. (sks May 25 98)
*/
/*****
*****/
int get_env_vars(void) {
    if (strcpy(env_tpcd_dbname,
getenv("TPCD_DBNAME")) == NULL) {
        fprintf(stderr, "\n The environment variable
$TPCD_DBNAME is not setup correctly.\n");
        return -1;
    }
    if (strcpy(env_user, getenv("USER")) == NULL) {
        fprintf(stderr, "\n The environment variable
$USER is not setup correctly.\n");
        return -1;
    }
    if (strcpy(env_tpcd_audit_dir,
getenv("TPCD_AUDIT_DIR")) == NULL) {
        fprintf(stderr, "\n The environment variable
$TPCD_AUDIT_DIR is not setup correctly.\n");
        return -1;
    }
    if (strcpy(env_tpcd_tmp_dir,
getenv("TPCD_TMP_DIR")) == NULL) {
        fprintf(stderr, "\n The environment variable
$TPCD_TMP_DIR is not setup correctly.\n");
        return -1;
    }
}
#if 0
    if (strcpy(env_tpcd_path_delim,
getenv("TPCD_PATH_DELIM")) == NULL ||
        (strcmp(env_tpcd_path_delim, "/") &&
strcmp(env_tpcd_path_delim, "\\"))){
        fprintf(stderr, "\n The environment variable
$TPCD_PATH_DELIM is not setup correctly ,
env_tpcd_path_delim%s'.\n",
env_tpcd_path_delim);

        return -1;
    }
}
#endif
    strcpy( env_tpcd_path_delim , "/" ); /*kmw*/
    if (strcpy(env_tpcd_run_on_multiple_nodes,
getenv("TPCD_RUN_ON_MULTIPLE_NODES"))
== NULL) {
        fprintf(stderr, "\n The environment variable
$TPCD_RUN_ON_MULTIPLE_NODES");
        fprintf(stderr, "\n is not setup correctly.\n");
        return -1;
    }
    if (strcpy(env_tpcd_copy_dir,
getenv("TPCD_COPY_DIR")) == NULL) {
        fprintf(stderr, "\n The environment variable
$TPCD_COPY_DIR is not setup correctly.\n");
        return -1;
    }
}

```

```

/* If TPCD_UPDATE_IMPORT is not set then,
the default is set to false, */
/* which is done in init_setup subroutine
*/
    strcpy(env_tpcd_update_import,
getenv("TPCD_UPDATE_IMPORT"));

    return 0;
}
/*****
*****/
/* Get the SQL statement and any control statements
from input. */
/*****
*****/
int Get_SQL_stmt(struct global_struct *g_struct)
{
    char input_ln[256] = "\0"; /* buffer for 1 line
of text */
    char temp_str[4000] = "\0"; /* temp string for
SQL stmt */
    char control_str[256] = "\0"; /* control string
*/

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

    int good_sql = 0; /* good-sql stmt flag
@d23684 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 (ostream, "\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 tjt */
        if (strstr(input_ln,"--#EOBLK") == input_ln)
        {
            g_struct->c_flags->eo_block = 1;
            return TPCDBATCH_EOBLOCK;
        }
        if (strstr(input_ln, "-- Query") == input_ln)
            strcpy(g_struct->s_info_ptr-
>qry_description,input_ln);

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

        /* if we're using update functions, return that
info
appropriately */
        if (g_struct->c_l_opt->update != 0) {

```

```

    if (strstr(input_ln, "--#INSERT") ==
input_ln)
        return TPCDBATCH_INSERT;

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

    if (strstr(input_ln, "--#COMMENT") ==
input_ln) { /* @d25594 tjt */
        temp_ptr = (input_ln + 11); /* User-
specified comments go to
the outfile */

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

        eostmt=0;
    }

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

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

```

```

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

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

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

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

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

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

    } while (!eostmt);

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

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

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

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

    else /* @d23684
tjg */
        return TPCDBATCH_NONSQL;
    } /* Get_SQL_stmt */

    /**
    *****
    *****
    /* allocate_sqllda -- This routine allocates space for
    the SQLDA. */
    /**
    *****
    *****
    void allocate_sqllda(struct sqllda *sqllda)
    {
        int loopvar; /* Loop counter
        */

        for (loopvar=0; loopvar<sqllda->sqlld; loopvar++)
        {
            switch (sqllda->sqlvar[loopvar].sqltype)
            {
                case SQL_TYP_INTEGER: /*
INTEGER */
                case SQL_TYP_NINTEGER:
                    if ((sqllda->sqlvar[loopvar].sqldata=
                    (TPCDBATCH_CHAR
                    *)malloc(sizeof(sqlint32))) == NULL)
                        mem_error("allocating INTEGER");
                    break;
                case SQL_TYP_BIGINT: /*
BIGINT */ /*kmwBIGINT*/

```

```

    case SQL_TYP_NBIGINT:
/*#ifdef SQLWINT */
/*   if ((sqlda->sqlvar[loopvar].sqldata= */
/*       (TPCDBATCH_CHAR
*)malloc(sizeof(__int64))) == NULL)*/
/* #else */
    if ((sqlda->sqlvar[loopvar].sqldata=
        (TPCDBATCH_CHAR
*)malloc(sizeof(sqlint64))) == NULL)
/* #endif*/
        mem_error("allocating BIGINT");
        break;
    case SQL_TYP_CHAR: /*
CHAR */
    case SQL_TYP_NCHAR:
    if ((sqlda->sqlvar[loopvar].sqldata=
        (TPCDBATCH_CHAR
*)calloc(256,sizeof(char))) == NULL)
        mem_error("allocating
CHAR/VARCHAR");
        break;
    case SQL_TYP_VARCHAR: /*
VARCHAR */
    case SQL_TYP_NVARCHAR:
    if ((sqlda->sqlvar[loopvar].sqldata=
        (TPCDBATCH_CHAR
*)calloc(4002,sizeof(char))) == NULL)
        mem_error("allocating
CHAR/VARCHAR");
        break;
    case SQL_TYP_LONG: /*
LONG VARCHAR */
    case SQL_TYP_NLONG:
    if ((sqlda->sqlvar[loopvar].sqldata=
        (TPCDBATCH_CHAR
*)calloc(32702,sizeof(char))) == NULL)
        mem_error("allocating VARCHAR/LONG
VARCHAR");
        break;
    case SQL_TYP_FLOAT: /*
FLOAT */
    case SQL_TYP_NFLOAT:
    if ((sqlda->sqlvar[loopvar].sqldata=
        (TPCDBATCH_CHAR
*)malloc(sizeof(double))) == NULL)
        mem_error("allocating FLOAT");
        break;
    case SQL_TYP_SMALL: /*
SMALLINT */
    case SQL_TYP_NSMALL:
    if ((sqlda->sqlvar[loopvar].sqldata=
        (TPCDBATCH_CHAR
*)malloc(sizeof(short))) == NULL)
        mem_error("allocating SMALLINT");
        break;

```

```

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

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

```

```

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

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

    int col_type;     /* Type of column
*/

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

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

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

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

                    fprintf(outstream, "%*ld ",col_lengths[col],
*(sqlint32 *)sqlda-
>sqlvar[col].sqldata));
                    break;

                case SQL_TYP_BIGINT: /*kmwBIGINT*/
                case SQL_TYP_NBIGINT:
/*#ifdef SQLWINT*/
/*          fprintf(outstream, "%*I64d
",col_lengths[col],*/
/*          *(__int64 *)sqlda-
>sqlvar[col].sqldata);*/
/*#else*/

                    fprintf(outstream, "%*ld ",col_lengths[col],
*(sqlint64 *)sqlda-
>sqlvar[col].sqldata));
/*#endif*/
                    break;

                case SQL_TYP_CHAR:
                case SQL_TYP_NCHAR:

                    fprintf(outstream, "%-*s
",col_lengths[col],sqlda->sqlvar[col].sqldata);
                    break;
                case SQL_TYP_VARCHAR:
                case SQL_TYP_NVARCHAR:
                case SQL_TYP_LONG:
                case SQL_TYP_NLONG:
/* @d30369 tjj */
                    ((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_NSMAIL:

                    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].sqlen).m;

```

```

        n=(*(struct declen *)&sqlda-
>sqlvar[col].sqlen).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].sqlen+1]='\0';
            strcpy(temp_string,(char *)sqlda-
>sqlvar[col].sqldata);
            fprintf(outstream, "%-*s
", (col_lengths[col]),temp_string);
            break;

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

    fprintf(outstream, "\n");

    return;
}

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

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

#ifdef (SQLOS2) || defined (SQLWINT) ||
defined (SQLWIN) || defined (SQLDOS)
    /*@d33143aha*/
    ftime (start_time);
#elif defined(SQLSNI)
    rc = gettimeofday(start_time);
#elif defined(SQLPTX)
    gettimeofday_mapped(start_time);
    rc = 0; /* gettimeofday_mapped returns void */
#elif defined (SQLUNIX) || defined (SQLAIX)
    /*TIMER jen*/
    rc = gettimeofday(start_time,NULL);
#else
#error Unknown operating system
#endif

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

/*****
*****/
/* Calculate and return the elapsed time given a
starting time. */
/*****
*****/

double get_elapsed_time ( Timer_struct *start_time)
{
    int status = 0;
    Timer_struct end_time;
    double result = -1.0;
#ifdef SQLWINT
    long int result_sec;
    long int result_usec;
#endif
}

#ifdef (SQLSNI)

```



```

    status = gettimeofday(&end_time);
#elif defined(SQLPTX)
    gettimeofday_mapped(&end_time);
    status = 0; /* gettimeofday_mapped returns void
*/
#elif defined (SQLUNIX) || defined (SQLAIX)
    status = gettimeofday(&end_time,NULL);
/*TIMER jen*/
#elif defined (SQLOS2) || defined (SQLWINT) ||
defined (SQLWIN) || defined(SQLDOS)
    ftime(&end_time);
#else
    /** If another operating
system **/
#error Unknown operating system
#endif

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

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

    }

/*
 * translate the time to that rounded to the
CLOSEST 0.1 seconds as
 * required by the TPC-D spec. ROUNDING
 */

```

```

    /* result = (double)(((long)((result + 0.099999) *
10))/10.0);*/
    result = (double)(((long)((result + 0.05) *
10))/10.0);
    return (result);
}

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

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

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

```

```

    struct sqlca temp_sqlca; /* temporary sqlca */
/* @d30369 tjt */

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

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

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

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

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

        EXEC SQL CONNECT ;
/* @d28763 tjt */

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

```

```

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

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

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

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

    return (string);
}

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

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

```

```

return (string);
}

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

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

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

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

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

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

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

```

```

break;

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

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

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

break;

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

```

```

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

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

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

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

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

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

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

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

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

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

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

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

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

#ifdef PARALLEL_UPDATES
    if (running_updates) {
        if (updatePair == -1) {
            fprintf (stderr, "The parameters to tpcdbatch
have not been passed correctly\n");
            exit (-1);
        }
        else {
            /* check to see if we are to use copy on for the
load */
            if (( getenv("TPCD_LOG") != NULL ) &&
(!strcmp(uppercase(getenv("TPCD_LOG")), "YES")
))
                {

```

```

        /* okay, we have set LOG_RETAIN on so
we need to use copy directory */
        copyOnOrOff = TRUE;
    }
    else
    {
        /* log retain off don't use copy directory */
        copyOnOrOff = FALSE;
    }

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

/* If no database name is given, then use the one
specified in the
environment variable DB2DBDFT, otherwise
error */
if (!strcmp(dbname, "\0")) {
    testptr = getenv("DB2DBDFT");
    if (testptr == NULL) {
        fprintf(stderr, "\nNo database name has been
specified on command ");
        fprintf(stderr, "line\n\nor in environment
variable DB2DBDFT.");
        display_usage();
    }
    else
        strcpy(dbname, testptr);
}
/* kjd715 */
/*
if (g_struct->c_l_opt->outfile) &&

```

```

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

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

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

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

    limit[ 0 ] = scal; limit[ 1 ] = prec - scal; limit[ 2 ] =
0;
    src_nibble = LEFT;
    for( j = 0 ; j < 2 ; j++ )
    {
        for( count = limit[ j ] ; count > 0 ; count-- )
        {

```

```

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

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

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

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

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

void init_setup(int argc, char *argv[], struct
global_struct *g_struct)
{
    int connect=0;
#ifdef SQLWINT
    char *pid;
#endif
    char temparray[256]="\0";
    int loopvar=0;

FILE *updateFP;
FILE *fpSeed;
char file_name[256] = "\0";
short seedEntry;
long lSeed;
int i;

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

/*****
*****
/* Start the mainline report processing.
*/
/*****
*****
if (!strcmp(g_struct->c_l_opt->infile, "\0")) {
    instream=stdin;
}
else {
    instream=NULL;
    if ( (instream = fopen(g_struct->c_l_opt->infile,
READMODE)) == NULL ) {
        /* kjd715 */
        fprintf(outstream, "XXThe input file could not
be opened.\n\n");
        /* kjd715 */
        fprintf(stdout, "Make sure that the filename is
correct.\n");
        fprintf(stdout, "filename = %s\n", g_struct-
>c_l_opt->infile);
        exit(-1);
    } /* open the input file if specified */
}

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

```

```

/* IMPORT (end) */

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

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

```

```

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

sprintf(file_name, "%s%sauditruns%smseedme",
env_tpcd_audit_dir,
env_tpcd_path_delim,
env_tpcd_path_delim);

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

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

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

```



```

/* Make sure that DB2 is started.
*/
/* CONNECT now unless this is a UF stream for a
Throughput test. */
/* (aph 98/12/22) */
/*****
*****/

if (g_struct->c_l_opt->update > 1)
{
/* This is an update function stream in a
throughput run. */
/* Just make sure that DB2 is started. Each UF
child will CONNECT itself. */
if (verbose) fprintf(stderr, "\nStarting the DB2
Database Manager Now\n");
sqlistar ();
}
else
{ /* In all other cases, CONNECT to the target
database. */
do
{
if (!strcmp(userid, "\0")) /** No authentication
provided **/
EXEC SQL CONNECT TO :dbname;
else EXEC SQL CONNECT TO :dbname
USER :userid USING :passwd;
if (sqlca.sqlcode == SQLE_RC_NOSTARTG)
{
if (verbose)
fprintf(stderr, "\nStarting the DB2 Database
Manager Now\n");
sqlistar ();
connect=0;
}
else connect=1;
} while (!connect);
error_check();
}

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

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

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

```

```

get_time_stamp(T_STAMP_FORM_2, &(g_struct-
>stream_start_time)); /* TIME_ACC jen*/

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

/* if we are running a throughput test, then we
must report the */
/* stream count information...we will report one
file per stream */
/* and amalgamate them after all streams have
completed */
/* if the number of streams is greater than 0 then
this is a throughput test*/
switch (g_struct->c_l_opt->update)
{
case (2):
case (5):
/* update throughput function stream */
sprintf(file_name, "%s%sstrcntuf.%s", g_struct-
>run_dir,
env_tpcd_path_delim, g_struct-
>file_time_stamp);
break;
case (3):
case (4):
/* update power function stream */
sprintf(file_name, "%s%spsprcntuf.%s", g_struct-
>run_dir,
env_tpcd_path_delim, g_struct-
>file_time_stamp);
break;
case (1):
/* power query stream */
sprintf(file_name,
"%s%spsprcnt%d.%s", g_struct->run_dir,
env_tpcd_path_delim,
g_struct->c_l_opt-
>intStreamNum, g_struct->file_time_stamp);
break;
case (0):
/* throughput query stream */
sprintf(file_name,
"%s%sstrcnt%d.%s", g_struct->run_dir,
env_tpcd_path_delim,
g_struct->c_l_opt-
>intStreamNum, g_struct->file_time_stamp);
break;
}

```

```

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

    if (g_struct->c_l_opt->update > 1)
    {
        /* update function stream */
        fprintf(g_struct->stream_report_file,
            "Update function stream starting at
%*.*s\n",
            T_STAMP_3LEN, T_STAMP_3LEN, /*
TIME_ACC jen*/

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

get_time_stamp(T_STAMP_FORM_3, &(g_struct-
>stream_start_time)); /* TIME_ACC jen*/
    }

#ifdef LINUX

    fclose(g_struct->stream_report_file);

#endif

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

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

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

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

    return ;
}

```

```

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

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

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

    fprintf (outstream, "\n");

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

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

        switch (col_type)
        {
            case SQL_TYP_SMALL:
            case SQL_TYP_NSMALL:
/* @d30369 tjc */
                col_lengths[col] = TPCDBATCH_MAX
(col_name_length,6);
                break;
            case SQL_TYP_INTEGER:
            case SQL_TYP_NINTEGER:

```

```

                col_lengths[col] = TPCDBATCH_MAX
(col_name_length,11);
                break;
            case SQL_TYP_BIGINT: /*kmwBIGINT*/
            case SQL_TYP_NBIGINT:
                col_lengths[col] = TPCDBATCH_MAX
(col_name_length,19);
                break;
            case SQL_TYP_CSTR:
            case SQL_TYP_NCSTR:
            case SQL_TYP_DATE:
            case SQL_TYP_NDATE:
            case SQL_TYP_TIME:
            case SQL_TYP_NTIME:
            case SQL_TYP_STAMP:
            case SQL_TYP_NSTAMP:
            case SQL_TYP_CHAR:
            case SQL_TYP_NCHAR:
            case SQL_TYP_VARCHAR:
            case SQL_TYP_NVARCHAR:
            case SQL_TYP_LONG:
            case SQL_TYP_NLONG:
                col_lengths[col] = TPCDBATCH_MAX
(col_name_length,col_width);
                break;

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

            case SQL_TYP_DECIMAL:
            case SQL_TYP_NDECIMAL:

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

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

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

```

```

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

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

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

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

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

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

    if ((form == T_STAMP_FORM_1) || (form ==
T_STAMP_FORM_3))
    {
        /* SUN fix bbe start */
#if (defined (SQLWINT) || defined (SQLWIN) ||
defined (SQLOS2) || defined (SQLDOS))
            timeLength = strftime(newtime,50,"%x %X",tp);
#elif (defined (SQLUNIX) || defined (SQLAIX))
            timeLength = strftime(newtime,50,"%D
%T",tp); /* SUN ...test this */
#else
#error Unknown operating system

```

```

#endif
    /* SUN fix bbe stop */
    /* TIME_ACC jen start*/
    if (form == T_STAMP_FORM_3)
    {
        /* concatenate the microsecond/milliseconds
on the end of the */
        /*timestamp jen1006 */
#if defined (SQLUNIX) || defined (SQLAIX)

        sprintf(newtime+timeLength,"%0.6d",temp_stamp.t
v_usec);
#elif (defined (SQLOS2) || defined (SQLWINT) ||
defined (SQLWIN) || defined (SQLDOS))

        sprintf(newtime+timeLength,"%0.3d",temp_stamp.
millitm);
#else
#error Unknown operating system
#endif
    /* TIME_ACC jen stop*/
    }
    else
        if (form == T_STAMP_FORM_2)
            strftime(newtime,50,"%y%m%d-
%H%M%S",tp);

    return (newtime);
}

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

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

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

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

```

```

double Ts2;
/* double QppD = 0;           MARK
double QthD = 0;
double QphD = 0; */

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

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

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

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

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

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

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

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

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

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

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

```

```

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

    s_info_ptr = s_info_ptr->next;

}

s_info_ptr = s_info_head_ptr;

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

    if (s_info_ptr->elapse_time != -1) {
        s_info_ptr->adjusted_time = s_info_ptr-
>elapse_time;
        /* determine whether the elapsed times have to
be adjusted or not */
        /* if this is an update function, we do not
adjust the elapsed time*/
        if ( strstr(s_info_ptr->tag,"UF") == NULL )
        {
            /* this is not an update function, adjust time
if necessary */
            if (max->elapse_time/min->elapse_time >
1000)
            {
                /* jmc fix geo_mean calculation...round
adjusted time properly ROUNDING*/
                adjusted_max_time = max-
>elapse_time/1000;
                if (s_info_ptr->elapse_time <
adjusted_max_time)
                {
                    s_info_ptr->adjusted_time =
(double)(((long)((adjusted_max_time +
0.05) * 10))/10.0);
                    if (s_info_ptr->adjusted_time < 0.1)
                        s_info_ptr->adjusted_time = 0.1;
                }
            }
        }
    }
}

```

```

    /*jmc fix geo_mean calculation...round
adjusted time properly ROUNDING end*/
    }
}

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

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

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

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

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

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

```

```

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

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

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

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

```

```

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

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

#ifdef (SQLUNIX) || defined (SQLAIX)
        Ts = g_struct->stream_end_time.tv_sec -
g_struct->stream_start_time.tv_sec + 1;
        Ts1 = (double)g_struct-
>stream_start_time.tv_sec + ((double)g_struct-
>stream_start_time.tv_usec/1000000);
        Ts2 = (double)g_struct-
>stream_end_time.tv_sec + ((double)g_struct-
>stream_end_time.tv_usec/1000000);
#elif (defined (SQLOS2) || defined (SQLWINT) ||
defined (SQLWIN) || defined (SQLDOS))
        Ts = g_struct->stream_end_time.time - g_struct-
>stream_start_time.time + 1;
        Ts1 = (double)g_struct->stream_start_time.time
+ ((double)g_struct-
>stream_start_time.millitm/1000);
        Ts2 = (double)g_struct->stream_end_time.time
+ ((double)g_struct-
>stream_end_time.millitm/1000);
#else
#error Unknown operating system
#endif

        /* TIME_ACC jen stop*/

        /* MARK
###Now do in calcmetrics.pl###

```

```

    QppD = (3600 * g_struct->scale_factor) /
adjusted_g_mean;
    QthD = (num_stmt * 3600 * g_struct-
>scale_factor) / Ts;
    QphD = sqrt(QppD*QthD);
    /*
    /* if the decimal part has some meaningful value
then print the database size
    with decimal part; otherwise just print the
integer part */

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

/*****
*****/
/* free up all the elements of the sqllda after done
processing */
/*****
*****/
void free_sqllda (struct sqllda *sqllda, int
select_status) /* @d30369 tjt */
{
    int loopvar;

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

    free(sqllda);
    sqllda_allocated = 0; /* fix free() problem on NT
wlc 090597 */
}

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

    char sourcedir[256];

    int split_updates = 2; /* no. of ways update
records are split */
    int concurrent_inserts = 2; /* jenCI no of
concurrent updates to be */
        /* jenCI run at once*/
    int loop_updates = 1; /* jenCI no of updates to
be run in one */
        /* jenCI "concurrent"
invocation. should*/
        /* jenCI be split_updates /
concurrent_inserts*/
    int i;
    int streamNum;
#ifdef SQLWINT
    /* PROCESS_INFORMATION
childprocess[100]; */
    char cmdline[256];
    HANDLE su_hSem;
    char UF1_semfile[256];
#else
    int childpid[100];
    int su_semid; /* semaphore for
controlling split updates*/
    key_t su_semkey; /* key to generate
semid */
#endif
    if (g_struct->c_l_opt->intStreamNum == 0)
        streamNum = 0;
    else
        streamNum = currentUpdatePair -
updatePairStart + 1;

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

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

    /* Load the orderkeys into the staging table */
    /* In SMP environments one could use a load
command but by using a */
    /* script we can keep the code common */
#ifdef SQLWINT
    sprintf (statement, "perl %s\\tools\\ploaduf1
%d\n", sourcedir, updatePair);
#else

```



```

    sprintf(statement, "perl %s/tools/ploaduf1 %d 1",
sourcedir, updatePair);
#endif
    if (system(statement))
    {
        fprintf(stderr, "ploaduf1 failed for UF1, examine
UF1.log for cause. Exiting.\n");
        if (verbose)
            fprintf(stderr,
                "ploaduf1 failed for UF1, examine UF1.log
for cause. Exiting.\n");
        exit (-1);
    }

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

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

#ifdef SQLWINT
    /* we will use the tpcd.setup file to generate the
semaphore key */
    if (getenv("TPCD_AUDIT_DIR") != NULL)
/*begin SEMA */
    {
        /* this is assuming that you will be running this
from 0th node */
        sprintf(sourcefile, "%s%ctools%ctpcd.setup",
            getenv("TPCD_AUDIT_DIR"),
PATH_DELIM,PATH_DELIM);
    }
    else
    {
        fprintf(stderr, "runUF1 Can't open UF1
semaphore file,TPCD_AUDIT_DIR is not
defined.\n");
        exit (-1);
    }
/*end SEMA */
    su_semkey = ftok (sourcefile, 'J');
    if ((su_semid = semget (su_semkey, 1,
IPC_CREAT|S_IRUSR|S_IWUSR)) < 0)
    {
        fprintf(stderr, "Cannot get semaphore! semget
failed: errno = %d\n",errno);
        exit (-1);
    }
#else /* SQLWINT */
    sprintf (UF1_semaphore, "%s.%s.UF1.semfile",
env_tpcd_dbname, env_user);
    su_hSem = CreateSemaphore(NULL, 0,
        concurrent_inserts,
/*jenCI*/
        (LPCTSTR)(UF1_semaphore));
    if (su_hSem == NULL)
    {
        fprintf(stderr,
            "CreateSemaphore (ready semaphore)
failed, GetLastError: %d, quitting\n",
            GetLastError());
        exit(-1);
    }
#endif /* SQLWINT */
    if (verbose) fprintf(stderr, "Semaphore created
successfully!\n");

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

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

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

    /* All children have been created, now wait for
them to finish */
#ifdef SQLWINT

```

```

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

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

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

}

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

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

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

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

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

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

    /* How many "chunks" have the orderkeys been
divided into? */

```

```

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

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

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

#ifdef SQLWINT
    sprintf (statement, "perl %s\\tools\\ploaduf2
%d\\n", sourcedir, updatePair);
#else
    sprintf (statement, "perl %s/tools/ploaduf2 %d 2",
sourcedir, updatePair);
#endif
    if (system(statement))
    {
        fprintf (stderr, "ploaduf2 failed for UF2, examine
UF2.log for cause. Exiting.\\n");
        exit (-1);
    }
    fprintf (outstream, "ploaduf2 finished for
UF2.\\n");

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

    /* Next we need to get ready to launch a bunch of
concurrent processes */
#ifdef SQLWINT
    /* we will use the tpcd.setup file to generate the
semaphore key begin SEMA */
    if (getenv("TPCD_AUDIT_DIR") != NULL)

```

```

    {
        sprintf(sourcefile, "%s%ctools%ctpcd.setup",
getenv("TPCD_AUDIT_DIR"),
PATH_DELIM, PATH_DELIM);
    }
    else
    {
        fprintf (stderr, "runUF2 Can't open UF2
semaphore file, TPCD_AUDIT_DIR is not
defined.\\n");
        exit (-1);
    }

    su_semkey = ftok (sourcefile, 'D'); /* use D for
deletes */
    /* end SEMA */
    if ( (su_semid = semget (su_semkey, 1,
IPC_CREAT|S_IRUSR|S_IWUSR)) < 0)
    {
        fprintf (stderr, "UF2 Can't get semaphore!
semget failed: errno = %d\\n",
errno);
        exit (-1);
    }
#else
    sprintf (UF2_semfile, "%s.%s.UF2.semfile",
env_tpcd_dbname, env_user);
    fprintf(stderr,"UF2 semfile = %s\\n",UF2_semfile);
    su_hSem = CreateSemaphore(NULL, 0,
concurrent_deletes,
(LPCTSTR)(UF2_semfile));
    if (su_hSem == NULL)
    {
        fprintf(stderr,
"CreateSemaphore (ready semaphore)
failed, GetLastError: %d, quitting\\n",
GetLastError());
        exit(-1);
    }
    fprintf(stderr,"Semaphore created
successfully!\\n");
#endif

    for (i=0; i < concurrent_deletes; i++)
    {
#ifdef SQLWINT
        if ((childpid[i] = fork()) == 0)
        {
            fprintf(stderr, "B-Calling runUF2_fn %d %d
%d ...\\n",
updatePair,
i,chunks_per_concurrent_delete);
            /* runUF2_fn (updatePair, i,
chunks_per_concurrent_delete); aph 981205 */

```

```

        runUF2_fn (updatePair, i,
chunks_per_concurrent_delete, dbname, userid,
passwd);
    }
    else
    {
        /* This is the parent */
        if (verbose)
            fprintf (stderr, "stream #%%d started with pid
%d\n", i, childpid[i]);
    }
#else
    {
        /* SECURITY_ATTRIBUTES sec_process;
SECURITY_ATTRIBUTES sec_thread; */
        /* NEED TO FIX THIS UP - KBS 98/10/20 */

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

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

```

```

//        fprintf(stderr,"About to wait again ...Sets to
wait for %d\n",
//            split_deletes - i);
        fprintf(stderr,"About to wait again ...Sets to
wait for %d\n",
            concurrent_deletes - i);
    }
    if (WaitForSingleObject(su_hSem, INFINITE)
== WAIT_FAILED)
    {
        fprintf(stderr,
            "WaitForSingleObject (su_hSem) failed
on set %d, error: %d, quitting\n",
            i, GetLastError());
        exit(-1);
    }
}
if (! CloseHandle(su_hSem))
{
    fprintf(stderr, "Close Sem failed - Last Error:
%d\n", GetLastError());
    /* no exit here */
}
#endif

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

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

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

/*-----
-*/
/*      General semaphore function.
*/
/*-----
-*/
#ifdef SQLWINT
int sem_op (int semid, int semnum, int value)
{

```

```

    struct sembuf sembuf; /* = {semnum ,value,0};
*/
    sembuf.sem_num = semnum;
    sembuf.sem_op = value;
    sembuf.sem_flg = 0;

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

/*****
*****
*/
/* Determines the proper name for the output file to
be generated for a particular TPC-D query, update
function, or
interval summary */
/*****
*****
*/
void output_file(struct global_struct *g_struct)
{
    char file_name[256] = "\0";
    char run_dir[150] = "\0";
    char time_stamp[50] = "\0";
    char delim[2] = "\0";
    int qnum=0, found=0; /* kjd715 */
    char input_ln[256] = "\0"; /* kjd715 */
    char tag[128] = "\0"; /* kjd715 */

    strcpy(run_dir,g_struct->run_dir);
    sprintf(delim,"%s",env_tpcd_path_delim);
    strcpy(time_stamp,g_struct->file_time_stamp);
    /* kjd715 */
    if (g_struct->stream_list == NULL)
    {
        if((g_struct->stream_list =
fopen(g_struct->c_l_opt->infile,
README)) == NULL)
        {
            fprintf(stderr,"\nThe input file could not be
opened.");
            fprintf(stderr,"Make sure that the filename is
correct.\n");
            exit(-1);
        }
    }
    found = 0;
    do {
        fscanf(g_struct->stream_list, "\n%[^\\n]\n",
input_ln);

```

```

    if (strstr(input_ln, "--#TAG") == input_ln)
    {
        found = 1;
        strcpy(tag,(input_ln+sizeof("--#TAG")));
        if(strncmp(tag, "UF", 2) == 0)
            qnum = atoi(tag+2)*(-1);
        else if(strncmp(tag, "Q", 1) == 0 )
        {
            /* for query 15a the 'a' must be
trimmed */
            /* off before converting to integer
*/
            if(strlen(tag)>3)
                tag[3] = '\0';
            qnum = atoi(tag+1);
        }
    }

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

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

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

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

```

```

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

}

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

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

if (!feof(instream) || g_struct->c_flags->eo_infile)

```

```

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

return;
}

/*****
*****/
/* Determine whether or not we should break out of
the block loop
because of an end of file, end of block, or update
function.
Also handle some semaphore stuff for update
functions */
/*****
*****/
int PreSQLprocess(struct global_struct *g_struct,
Timer_struct *start_time)
{
int rc = 1;
FILE *updateFP;
#ifdef SQLWINT
int semid; /* semaphore for
controlling UFs*/
key_t semkey; /* key to generate
semid */
#else
int SemTimeout = 60000; /* Des
time out period of 1 minute */
#endif

switch (g_struct->c_flags->select_status)
{
case TPCDBATCH_NONSQL:
g_struct->s_info_stop_ptr = g_struct-
>s_info_ptr;
/* if we're at the end of the input file, set the stop
pointer to this structure */
rc = FALSE;
break;
case TPCDBATCH_EOBLOCK:
rc = FALSE;
break;
case TPCDBATCH_INSERT:

```

```

    /* we have to check whether or not this is a
throughput */
    /* test, and if it is, we have to set up a semaphore
to */
    /* control when the update functions are run.
We want */
    /* them to be run after all the query streams have
finished. */
    /* What we do is set up the semaphore here,
decrement it */
    /* in the query streams, and wait for it to get
cleared */
    /* before we allow the UFs to run.
*/
    /* Note: we only set up the semaphore if:
*/
    /* 1. we are running the throughput test
(num of */
    /* streams > 0) */
    /* 2. we are at the first UF1 (i.e. this is the
*/
    /* case where currentUpdatePair =
updatePairStart */
    /* we also want to check the sem_on element in
the global */
    /* structure to see if we want to use semaphores
or let */
    /* the calling script do the synchronization of the
update */
    /* stream */
    if ( semcontrol == 1 )
    {
        /* yes we are to be using semaphores */
        /* is this the 1st time into update function 1
(uf1)? */
        if (currentUpdatePair == updatePairStart )
        {
            /* create the semaphores */
            create_semaphores(g_struct);
            if (g_struct->c_l_opt->intStreamNum !=
0)
            /* wait period for runthroughput updates
*/
                throughput_wait(g_struct);
        }
        /* otherwise continue to run*/
    }
    if ((g_struct->c_l_opt->update == 3) || (g_struct-
>c_l_opt->update == 4))
    {
        get_start_time(start_time);
        strcpy(g_struct->s_info_ptr->start_stamp,
get_time_stamp(T_STAMP_FORM_3,start_time ));
    /* TIME_ACC jen*/
    /* write the start timestamp to the file...if this is
not a qualification */
    /* run, then write the seed used as well */
    fprintf( ostream,"Start timestamp %*.*s \n",
T_STAMP_3LEN,T_STAMP_3LEN,
/* TIME_ACC jen*/
g_struct->s_info_ptr->start_stamp);
    if (g_struct->c_l_opt->intStreamNum >= 0)
    {
        if (g_struct->lSeed == -1)
        {
            fprintf( ostream,"Using default qgen seed
file");
        }
        else
            fprintf( ostream,"Seed used =
%ld",g_struct->lSeed);
        fprintf( ostream,"\n");
    }
    if (g_struct->c_l_opt->update < 4){
        /* run only if updates are enabled */
        runUF1(g_struct, currentUpdatePair);
    }

    rc = FALSE;
    if ((g_struct->c_l_opt->intStreamNum == 0) &&
(semcontrol == 1))
        /* RUNPOWER: release first semaphore so the
queries can run */
        release_semaphore(g_struct,
INSERT_POWER_SEM);
        break;
    case TPCDBATCH_DELETE:
        if ((g_struct->c_l_opt->intStreamNum == 0) &&
(semcontrol == 1))
        {
            /* RUNPOWER: wait for queries to finish */
            /* waiting on QUERY_POWER_SEM
semaphore */
            runpower_wait(g_struct,
QUERY_POWER_SEM);
        }
        if ((g_struct->c_l_opt->update == 3) || (g_struct-
>c_l_opt->update == 4))
        {
            get_start_time(start_time);
            strcpy(g_struct->s_info_ptr->start_stamp,
get_time_stamp(T_STAMP_FORM_3,start_time ));
    /* TIME_ACC jen*/
    /* write the start timestamp to the file...if this is
not a qualification */
    /* run, then write the seed used as well */
    fprintf( ostream,"Start timestamp %*.*s \n",

```

```

    /* write the start timestamp to the file...if this is
not a qualification */
    /* run, then write the seed used as well */
    fprintf( ostream,"Start timestamp %*.*s \n",
T_STAMP_3LEN,T_STAMP_3LEN,
/* TIME_ACC jen*/
g_struct->s_info_ptr->start_stamp);
    if (g_struct->c_l_opt->intStreamNum >= 0)
    {
        if (g_struct->lSeed == -1)
        {
            fprintf( ostream,"Using default qgen seed
file");
        }
        else
            fprintf( ostream,"Seed used =
%ld",g_struct->lSeed);
        fprintf( ostream,"\n");
    }
    if (g_struct->c_l_opt->update < 4){
        /* run only if updates are enabled */
        runUF1(g_struct, currentUpdatePair);
    }

    rc = FALSE;
    if ((g_struct->c_l_opt->intStreamNum == 0) &&
(semcontrol == 1))
        /* RUNPOWER: release first semaphore so the
queries can run */
        release_semaphore(g_struct,
INSERT_POWER_SEM);
        break;
    case TPCDBATCH_DELETE:
        if ((g_struct->c_l_opt->intStreamNum == 0) &&
(semcontrol == 1))
        {
            /* RUNPOWER: wait for queries to finish */
            /* waiting on QUERY_POWER_SEM
semaphore */
            runpower_wait(g_struct,
QUERY_POWER_SEM);
        }
        if ((g_struct->c_l_opt->update == 3) || (g_struct-
>c_l_opt->update == 4))
        {
            get_start_time(start_time);
            strcpy(g_struct->s_info_ptr->start_stamp,
get_time_stamp(T_STAMP_FORM_3,start_time ));
    /* TIME_ACC jen*/
    /* write the start timestamp to the file...if this is
not a qualification */
    /* run, then write the seed used as well */
    fprintf( ostream,"Start timestamp %*.*s \n",

```

```

        T_STAMP_3LEN,T_STAMP_3LEN,
/* TIME_ACC jen*/
        g_struct->s_info_ptr->start_stamp);
    if (g_struct->c_l_opt->intStreamNum >= 0)
    {
        if (g_struct->lSeed == -1)
        {
            fprintf( outstream,"Using default qgen seed
file");
        }
        else
            fprintf( outstream,"Seed used =
%ld",g_struct->lSeed);
        fprintf( outstream,"\n");
    }
}
if (g_struct->c_l_opt->update < 4){
/* run only if updates are enabled */
runUF2(g_struct, currentUpdatePair);
if (g_struct->c_l_opt->intStreamNum == 0)
{ /* RUNPOWER */
    fprintf(stderr, "UF2 completed\n");
}
}
currentUpdatePair += 1;
/* update the update.pair.num file to reflect the
successfully completed */
/* update pair */
if (g_struct->c_l_opt->update < 4)
{ /*jen*/
#ifdef NO_INCREMENT
    /* don't update the pair, only for my testing -
Haider */
    updateFP = fopen(g_struct-
>update_num_file,"w");
    fprintf(updateFP,"%d\n",currentUpdatePair);
    fclose(updateFP);
#endif
} /*jen*/
rc = FALSE;
break;

}
return(rc);
}

/*****
*****
*/
/* Handles actual processing of SQL statement.
Initializes the SQLDA
for returned rows, does PREPARE, DECLARE,
and OPEN statements and
executed multiple FETCHes as needed. If not a
SELECT statement,

```

```

    goes into EXECUTE IMMEDIATE section
*/
/*****
*****
*/
void SQLprocess(struct global_struct *g_struct)
{
    int rc = 0;          /* 912RETRY */
    int rows_fetch = 0;
    long sqlcode = SQL_RC_E911; /*
Temporary sqlcode to test
for deadlocks */
    int max_wait = 1;   /* Maximum
number of retries
for deadlock scenario
*/
    int col_lengths[TPCDBATCH_MAX_COLS];
/* array containing widths of
columns in returned
set */
    struct stmt_info *s_info_ptr;

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

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

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

    while (((sqlcode == SQL_RC_E911) ||
(sqlcode == SQL_RC_E912) ||
(sqlcode == SQL_RC_E901)) &&
(max_wait < MAXWAIT) &&
(rc==0) )
    {
        sqlcode = 0; /* Re-initialize sqlcode to
avoid infinite-loop */
        if (g_struct->c_flags->select_status ==
TPCDBATCH_SELECT)
        {
            /* Enter this loop if SQL stmt is a SELECT */
            EXEC SQL PREPARE STMT1 INTO :*sqlda
FROM :stmt_str;

```



```

        sqlcode = error_check();
        if (sqlcode < 0)
        {
            fprintf (stderr, "\nPrepare failed. Stopping this
query.\n");
            rc = -1;
        }
        else /* print out the column headings for the
answer set */
        {
            print_headings(sqllda,col_lengths);
/* @d22817 tlg */

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

            EXEC SQL DECLARE DYNCUR
CURSOR FOR STMT1;

            EXEC SQL OPEN DYNCUR;
            sqlcode = error_check();

            if (sqlcode < 0) /* we ran into an error of
some kind KBS 98/09/28 */
            {
                max_wait ++;
                fprintf (stderr, "\nAn error has been
detected on open...Retrying...\n");
                SleepSome(10);
            }
            else
            {

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

/******
*****
            rows_fetch = 0;

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

```

```

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

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

                if (sqlcode < 0 )
                {
                    max_wait ++;
                    fprintf (stderr, "\nAn error has been detected
on execute immediate...Retrying...\n");
                    SleepSome(10);
                }
            } /* end of block for handling NON-select
statements */

            if ( (sqlcode >= 0 ) &&
                (g_struct->c_flags->select_status ==
TPCDBATCH_SELECT))
            {

```

```

        /* we opened a cursor before */
        EXEC SQL CLOSE DYNCUR;
        sqlcode = error_check();

        if ((s_info_ptr->max_rows_fetch == -1) ||
            (rows_fetch < s_info_ptr-
>max_rows_fetch))
#ifdef SQLPTX
            fprintf (outstream, "\n\nNumber of rows
retrieved is: %6d",
                    rows_fetch);
        else
            fprintf (outstream, "\n\nNumber of rows
retrieved is: %6d",
                    s_info_ptr->max_rows_fetch);
#else
            fprintf (outstream, "\n\nNumber of rows
retrieved is: %6d",
                    rows_fetch);
        else
            fprintf (outstream, "\n\nNumber of rows
retrieved is: %6d",
                    s_info_ptr->max_rows_fetch);
#endif
    }
    /* @d28763
tjg */

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

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

} /* end of SQLprocess */

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

#ifdef DEBUG
    fprintf (outstream, "In PostSQLprocess\n");
#endif
#endif
    s_info_ptr = g_struct->s_info_ptr;

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

    if (g_struct->c_l_opt->update > 1)
    {
        /* This is an update function stream. There is no
need to COMMIT. */
        /* Each UF child will COMMIT its own
transactions. */
        ;
    }
    else
    { /* For non-UF cases, COMMIT now. */
        if (g_struct->c_l_opt->a_commit) {
            EXEC SQL COMMIT WORK;
            error_check(); /*
@d22275 tjg */
        }
    }

    fflush(outstream);

    s_info_ptr->elapse_time =
get_elapsed_time(start_time);

    if (g_struct->c_flags->time_stamp == TRUE)
/* @d25594 tjg */
        get_start_time(&end_t); /* Get the end time */
        strcpy(s_info_ptr->end_stamp,
get_time_stamp(T_STAMP_FORM_3,&end_t)
);

/*get_time_stamp(T_STAMP_FORM_3,(time_t)NU
LL);*/

/* BBE: Pass on time stamp values for the next
query */
    temp_time_struct = end_t;
    strcpy(temp_time_stamp, s_info_ptr->end_stamp);

/* write the start timestamp to the file */
    fprintf( outstream, "\n\nStop timestamp %*.s\n",
            T_STAMP_3LEN, T_STAMP_3LEN, /*
TIME_ACC jen*/
            s_info_ptr->end_stamp);

/* DJD print elapsed time in seconds */
    fprintf( outstream, "Query Time = %15.1f secs\n",
s_info_ptr->elapse_time);

```

```

    /** Allocate space for a new stmt_info structure
    **/ /* @d24993 tjt */
    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)
{
#ifdef SQLWINT
    int semid; /* semaphore for
controlling UFs*/
    key_t semkey; /* key to generate
semid */
#endif
    char file_name[256] = "\0";

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

    switch (g_struct->c_l_opt->update)
    {
        case (2):
        case (5):
            /* update throughput function stream */

            sprintf(file_name,"%s%sstrcntuf.%s",g_struct-
>run_dir,
                    env_tpcd_path_delim, g_struct-
>file_time_stamp);
                break;
        case (3):
        case (4):
            /* update power function stream */

            sprintf(file_name,"%s%spsprcntuf.%s",g_struct-
>run_dir,
                    env_tpcd_path_delim, g_struct-
>file_time_stamp);
                break;
        case (1):
            /* power query stream */
            sprintf(file_name,
"%s%spsprcnt%d.%s",g_struct->run_dir,
env_tpcd_path_delim,
                    g_struct->c_l_opt-
>intStreamNum,g_struct->file_time_stamp);
                break;
        case (0):
            /* throughput query stream */
            sprintf(file_name,
"%s%sstrcnt%d.%s",g_struct->run_dir,
env_tpcd_path_delim,
                    g_struct->c_l_opt-
>intStreamNum,g_struct->file_time_stamp);
                break;
    }
}

```

```

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

    /* print out the stream stop time in the stream
count information file*/
    if (g_struct->c_l_opt->update > 1)
    {
        /* update function stream */
        fprintf(g_struct->stream_report_file,
            "Update function stream stopping at
%*.*s\n",
            T_STAMP_3LEN, T_STAMP_3LEN, /*
TIME_ACC jen*/

get_time_stamp(T_STAMP_FORM_3, &(g_struct-
>stream_end_time))); /* TIME_ACC jen*/
    }
    else
    {
        /* query stream(s) */
        fprintf(g_struct->stream_report_file,
            "Stream number %d stopping at %*.*s\n",
            g_struct->c_l_opt->intStreamNum,
            T_STAMP_3LEN, T_STAMP_3LEN, /*
TIME_ACC jen*/

get_time_stamp(T_STAMP_FORM_3, &(g_struct-
>stream_end_time))); /* TIME_ACC jen*/
    }
    fclose(g_struct->stream_report_file);

    /* No need to check for errors here.
    Also, the UF stream in a Throughput run
    has no connection in tpcdbatch.sqc.      aph
98/12/26
    error_check();
    */

    /* if we are in a query stream AND this is a
throughput test, then need */
    /* do to some semaphore stuff (0 implies update
functions are off) */
    /* AND we are supposed to be using semaphores
*/
    if ( ( semcontrol == 1 ) &&
        ( g_struct->c_l_opt->update < 2))
        /* only queries need to release the semaphore at
this point */
        {
            if (g_struct->c_l_opt->intStreamNum == 0)
                release_semaphore(g_struct,
QUERY_POWER_SEM); /* power stream */
            else
                release_semaphore(g_struct,
THROUGHPUT_SEM); /* throughput stream */

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

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

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

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

    return (TRUE);
}

void create_semaphores(struct global_struct
*g_struct)
{
#endif SQLWINT
    int semid; /* semaphore for
controlling UFs*/
    key_t semkey; /* key to generate
semid */
#else
    HANDLE hSem;

```

```

HANDLE      hSem2;
int         SemTimeout = 600000; /* Des
time out period of 1 minute */
#endif
    fprintf(stderr,"numstreams = %d\n",g_struct-
>c_l_opt->intStreamNum);
    fprintf(stderr,"Update stream creating
semaphore(s) for update and query sequencing\n");
#ifdef SQLWINT

    fprintf(stderr,"semfile = %s\n",g_struct-
>sem_file);
    if (g_struct->c_l_opt->intStreamNum == 0)
        /*RUNPOWER*/
        {
            fprintf(stderr,"semfile2 = %s\n",g_struct-
>sem_file2);
            hSem = CreateSemaphore(NULL,
0,1,(LPCTSTR)(g_struct->sem_file));
            hSem2 = CreateSemaphore(NULL,
0,1,(LPCTSTR)(g_struct->sem_file2));
            if ((hSem == NULL) || (hSem2 ==
NULL))
            {
                fprintf(stderr,
                "CreateSemaphores (ready
semaphore) failed, GetLastError: %d, quitting\n",
                GetLastError());
                exit(-1);
            }
            fprintf(stderr,"Semaphores created
successfully!\n");
        }
    else
    {
        /* RUNTHROUGHPUT creates semaphores
based on the number of query streams while the
number of streams for runpower is constant */
        hSem = CreateSemaphore(NULL, 0,
g_struct->c_l_opt-
>intStreamNum,
                (LPCTSTR)(g_struct-
>sem_file));

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

```

```

#else      /* AIX, SUN, etc. */
    /* create a semaphore key...use the name of a
file that */
    /* you know exists */
    fprintf(stderr,"semfile = %s\n", g_struct-
>update_num_file);
    semkey = ftok(g_struct->update_num_file,'J');
    if (g_struct->c_l_opt->intStreamNum == 0)
        /* RUNPOWER */
        {
            if ( (semid =
semget(semkey,2,IPC_CREAT|S_IRUSR|S_IWUSR
)) < 0)
            {
                fprintf(stderr,
                "Throughput can't get initial
semaphore! semget failed errno = %d\n",
                errno);
                exit(1);
            }
        }
    else
        /* THROUGHPUT */
        {
            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));
            }
        }
}
#endif
}

/*throughput update */
void throughput_wait(struct global_struct *g_struct)
{
#ifdef SQLWINT

```

```

    int          semid;          /* semaphore for
controlling UFs*/
    key_t        semkey;        /* key to generate
semid */
#else
    HANDLE       hSem;
    int          j;
    int          SemTimeout = 60000; /* Des
time out period of 1 minute */
#endif

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

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

    if (sem_num == 1)
        strcpy (semfile, g_struct->sem_file);
    else
        strcpy (semfile, g_struct->sem_file2);
#else /* AIX */
    int          semid;          /* semaphore for
controlling UFs*/
    key_t        semkey;        /* key to generate
semid */

    strcpy (semfile, g_struct->update_num_file);
#endif

    if (g_struct->c_l_opt->update == 1)
        fprintf(stderr,"querystream waiting for update
stream (UF1) to signal semaphore based on %s\n",
semfile);
    else
        fprintf(stderr,"updatestream (UF2) waiting on
querystream semaphore to signal semaphore based
on %s\n", semfile);
#ifdef SQLWINT
    hSem = open_semaphore(g_struct, sem_num);
    if (verbose)
        fprintf(stderr,"Runpower queries about to wait
...\n");
        if (WaitForSingleObject(hSem, INFINITE) ==
WAIT_FAILED)
        {

```

```

        fprintf(stderr,
            "WaitForSingleObject (hSem) failed on
            stream 0, error: %d, quitting\n",
            GetLastError());
        exit(-1);
    }
    if (! CloseHandle(hSem))
    {
        fprintf(stderr, "Close Sem failed - Last
        Error: %d\n", GetLastError());
        /* no exit here */
    }
#else

    semid = open_semaphore(g_struct);

    /* call the sem_op routine to decrement the
    semaphore by */
    /* however many streams .... by calling this
    function with*/
    /* a negative number, this stream is forced to wait
    until */
    /* the semaphore gets back to 0 */
    /* aix semaphores start at 0, not 1, so sem_num -1
    is used */
    if (sem_op(semid, sem_num - 1, -1) != 0)
    {
        /*jenSEM*/
        fprintf(stderr,
            "Failure to wait on runpower semaphore
            for %d streams\n",
            g_struct->c_l_opt->intStreamNum);
        exit(1);
    }
    /*jenSEM*/
#endif
    if (g_struct->c_l_opt->update == 1)
        fprintf(stderr, "querystream finished waiting on
        updatestream semaphore\n");
    else
        fprintf(stderr, "updatestream finished waiting on
        querystream semaphore\n");
}

void release_semaphore(struct global_struct
*g_struct, int sem_num)
{
#ifdef SQLWINT
    int          semid;          /* semaphore for
    controlling UFs*/
    key_t        semkey;        /* key to generate
    semid */
#else
    HANDLE        hSem;
    int          SemTimeout = 60000; /* Des
    time out period of 1 minute */
#endif

```

```

#ifdef SQLWINT
    hSem = open_semaphore(g_struct, sem_num);
/* query */
    if (! ReleaseSemaphore(hSem,
        1,
        (LPLONG)(NULL)))
    {
        fprintf(stderr, "ReleaseSemaphore failed,
        Sem#: %d LastError: %d, quit\n",
        sem_num, GetLastError());
        exit(-1);
    }
#else
    semid = open_semaphore(g_struct); /* query */
    /* aix semaphores start at 0, not 1, so sem_num
    -1 is used */
    if (sem_op(semid, sem_num - 1, 1) != 0)
    /*jenSEM*/
    {
        /*jenSEM*/
        fprintf(stderr,
            "Failed to increment semaphore %d
            for throughput stream %d\n",
            sem_num, g_struct->c_l_opt-
            >intStreamNum);
        fprintf(stderr,
            "file for generation of semaphore is:
            %s\n",
            g_struct->update_num_file);
        exit(1);
    }
#endif
    if (g_struct->c_l_opt->intStreamNum == 0)
    { /* RUNPOWER */
        if (sem_num == 1)
        {
            fprintf(stderr, "UF1 completed.\n");
        }
        else
        {
            fprintf(stderr, "query stream completed.\n");
        }
    }
}

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

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

```

```

else
    semfile = (LPCTSTR)g_struct-
>sem_file2;

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

        Sleep(1000);
    }
    return hSem;
}

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

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

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

```

```

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

```

tpcdUF.sqc

```

/*****
*****
*
* TPCDUF.SQC
*
* Revision History:
*
* 05 dec 98 aph Created tpcdUF.sqc containing
runUF1_fn() and runUF2_fn()
* so that it can be bound separately with a
different isolation level.
* 15 may 99 bbe Added cast (short) for type
conversion between a long and a short.
* 16 jun 99 jen Added in proper connect reset code
for UF functions (mistakenly
* removed
* 17 jun 99 jen SEMA Changes semaphore file for
update functions to look for tpcd.setup
* not for the orders.*** update data file
(AIX only )
* 21 jul 99 bbe Commented out conditions in SQL
statments that searched on fields
* other than app_id.
*
*****
*****/

#define UF1DEBUG
#define UF2DEBUG

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

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

#include "sqlca.h"

```



```

extern struct sqlca sqlca;

/*****
*****
*/
/* Function Prototypes
*/
/*****
*****
*/
extern int SleepSome( int amount );
extern long error_check(void);
/* @d28763 tjg */
extern void dumpCa(struct sqlca*); /*kmw*/
extern int sem_op (int semid, int semnum, int value);
extern char *get_time_stamp(int form, Timer_struct
*timer_pointer); /* TIME_ACC jen */

/*****
*****
*/
/* Declare the SQL host variables.
*/
/*****
*****
*/
EXEC SQL BEGIN DECLARE SECTION;
char UF_dbname[9] = "\0";
char UF_userid[9] = "\0";
char UF_passwd[9] = "\0";
sqlint32 UF_chunk = 0;
short month = 0;
EXEC SQL END DECLARE SECTION;

/*****
*****
*/
/* Declare the global variables.
*/
/*****
*****
*/
extern char env_tpcd_tmp_dir[150];
extern FILE *instream, *outstream; /* File pointers
*/
extern char sourcefile[256]; /* Used for
semaphores and table functions?*/
extern struct { /* jen LONG */
short len;
char data[32700];
} stmt_str; /* jen LONG */

/*****
*****
*/
/* UF1 child */
/* (i is the application number.)
*/
/*****
*****
*/
void runUF1_fn ( int updatePair, int i, char
*dbname, char *userid, char *passwd )

```

```

{
int rc = 0;
int split_updates = 2; /* no. of ways update
records are split */
int concurrent_inserts = 2; /* jenCI no of
concurrent updates to be */
/* jenCI run at once*/
int loop_updates = 1; /* jenCI no of updates to
be run in one */
/* jenCI "concurrent"
invocation. should*/
/* jenCI be split_updates /
concurrent_inserts*/
int startChunk = 0; /* jenCI number of first
chunk to insert for */
/* jenCI this child */
int stopChunk = 0; /* jenCI number of last
chunk to insert for */
/* jenCI this child */
long insertedLineitem = 0; /*kmw*/
long insertedOrders = 0; /*kmw*/
long saveInsertedOrders = 0; /*kbs*/

long sqlcode;
int maxwait;

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

char myostreamfile[256];
FILE *myostream;

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

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

```

```

    ostream=myostream; /* initialize ostream
for error_check dxxxxhar*/

    fprintf( myostream, "\nUF1 for update pair %d
set %d starting at %*. *s\n",
            updatePair, i,
            T_STAMP_1LEN, T_STAMP_1LEN, /*
TIME_ACC jen*/

get_time_stamp(T_STAMP_FORM_1, (Timer_struct
t *)NULL)); /* TIME_ACC jen*/

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

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

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

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

```

```

for ( UF_chunk = startChunk; UF_chunk <=
stopChunk; UF_chunk++) /*jenCI*/
{
/*jenCI*/
    /* wlc 062797 */
    sqlcode = SQL_RC_E911;
    month = (short)UF_chunk; /* Cast 'short' added
bbe */
    maxwait = 1;
    rc = 0;

#ifdef UF1DEBUG
    fprintf (myostream, "Before While_a Chunk=
%d\n", UF_chunk);
    fflush(myostream);
#endif
    /* Loop to handle any deadlocks */
    while (sqlcode == SQL_RC_E911 && maxwait
<= MAXWAIT && rc==0)
    {
        sqlcode = 0;
#ifdef UF1DEBUG
        fprintf (myostream, "in loop before orders exec
sql\n");
        fflush(myostream);
#endif
        EXEC SQL INSERT INTO TPCD.ORDERS
        SELECT
O_ORDERKEY, O_CUSTKEY, O_ORDERSTATU
S, O_TOTALPRICE,
O_ORDERDATE, O_ORDERPRIORITY, O_CLER
K, O_SHIPPRIORITY, O_COMMENT
        FROM TPCDTEMP.ORDERS_NEW
        WHERE APP_ID = :UF_chunk;
        /*AND 12*(YEAR(O_ORDERDATE)-
1992)+MONTH(O_ORDERDATE)-01 = :month;*/

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

        if (sqlcode == SQL_RC_E911)
        {
            /* we've hit a deadlock */
            fprintf (myostream,
                    "\nDeadlock detected inserting from
tpcdtemp.orders_new for chunk %d for pair
%d..Retrying...\n", UF_chunk, updatePair);
            SleepSome(UF_DEADLOCK_SLEEP);
            maxwait++; /* jen
DEADLOCK */
        }
        else if (sqlcode < 0)
        {
            fprintf(myostream,
                    "Insert into orders pair %d chunk %d
failed sqlcode=%d\n",

```

```

        updatePair,UF_chunk,sqlcode);
    dumpCa(&sqlca);
    rc = -1;
}
else
{
    /* Everything worked with ORDERS,
    proceed with LINEITEM */
    saveInsertedOrders = sqlca.sqlerrd[2];

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

    EXEC SQL INSERT INTO
    TPCD.LINEITEM
    SELECT
    L_ORDERKEY,L_PARTKEY,L_SUPPKEY,L_LI
    NENUMBER,L_QUANTITY,

    L_EXTENDEDPRICE,L_DISCOUNT,L_TAX,

    L_RETURNFLAG,L_LINESTATUS,L_SHIPDAT
    E,L_COMMITDATE,L_RECEIPTDATE,

    L_SHIPINSTRUCT,L_SHIPMODE,L_COMMENT
    FROM TPCDTEMP.LINEITEM_NEW
    WHERE APP_ID = :UF_chunk;
    /*(AND L_ORDERKEY IN
    (SELECT O_ORDERKEY FROM
    TPCD.ORDERS
    WHERE
    12*(YEAR(O_ORDERDATE)-
    1992)+MONTH(O_ORDERDATE)-01 = :month);*/

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

    if (sqlcode == SQL_RC_E911)
    {
        /* we've hit a deadlock */
        fprintf (myostream,
                "\nA deadlock has been detected
        inserting from
        tpcdtemp.lineitem%d_%d...Retrying...\n",
                updatePair, UF_chunk);
        SleepSome(UF_DEADLOCK_SLEEP);
        maxwait++; /* jen
    DEADLOCK */
    }
    else if (sqlcode < 0)
    {
        fprintf(myostream,

```

```

        "Insert into lineitem pair %d chunk
        %d failed sqlcode=%d\n",
        updatePair,UF_chunk,sqlcode);
        dumpCa(&sqlca);
        rc = -1;
    }
    else
    {
#ifdef UF1DEBUG
        fprintf (myostream, "lineitem insert
        succeeded\n");
        fflush(myostream);
#endif
        /* accumulate the number of row inserted
        */
        /* Order count ONLY updated if both
        orders and lineitem */
        /* go through */
        insertedOrders += saveInsertedOrders;
    /* kbs */
        insertedLineitem += sqlca.sqlerrd[2];
        rc=0;
        EXEC SQL COMMIT WORK;
        error_check();

#ifdef UF1DEBUG
        /* report the number of row inserted */
        fprintf(myostream, " interim %ld rows
        for chunk %d into TPCD.ORDERS at %*.s\n",
        insertedOrders,UF_chunk,T_STAMP_1LEN,T_ST
        AMP_1LEN, /* TIME_ACC jen*/

        get_time_stamp(T_STAMP_FORM_1,(Timer_struc
        t *)NULL)); /* TIME_ACC jen*/
        /* report the number of row deleted *s
        inserted */
        fprintf(myostream,
                " interim %ld rows for chunk %d
        into TPCD.LINEITEM at %*.s\n",
                insertedLineitem,UF_chunk,
                T_STAMP_1LEN,T_STAMP_1LEN,
        /* TIME_ACC jen*/

        get_time_stamp(T_STAMP_FORM_1,
                (Timer_struct *)NULL)); /*
        TIME_ACC jen*/

        fprintf( myostream,
                " inserts for update pair %d chunk
        %d complete at %*.s\n\n",
                updatePair, UF_chunk,

        T_STAMP_1LEN,T_STAMP_1LEN, /*
        TIME_ACC jen*/

```

```

get_time_stamp(T_STAMP_FORM_1,
               (Timer_struct *)NULL)); /*
TIME_ACC jen*/
#endif
    }
    } /* process lineitem INSERTs */
    } /* while loop for deadlocks */
    } /* while processing chunks */

    /* report the number of row deleted */
    fprintf(myostream, "%ld rows inserted into
TPCD.ORDERS at %*.s\n",

insertedOrders,T_STAMP_1LEN,T_STAMP_1LEN
, /* TIME_ACC jen*/

get_time_stamp(T_STAMP_FORM_1,(Timer_struct
*)NULL)); /* TIME_ACC jen*/
    fprintf(myostream, "%ld rows inserted into
TPCD.LINEITEM at %*.s\n",

insertedLineitem,T_STAMP_1LEN,T_STAMP_1L
EN, /* TIME_ACC jen*/

get_time_stamp(T_STAMP_FORM_1,(Timer_struct
*)NULL)); /* TIME_ACC jen*/

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

        goto UF1_exit;
    }

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

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

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

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

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

#else /* SQLWINT */
    sprintf (UF1_semfile, "%s.%s.UF1.semfile",
            getenv("TPCD_DBNAME"),
            getenv("USER"));
    fprintf(stderr,"UF1 semfile = %s\n",UF1_semfile);
    while ((su_hSem =
OpenSemaphore(SEMAPHORE_ALL_ACCESS |
SEMAPHORE_MODIFY_STATE |
              SYNCHRONIZE,
              TRUE,
              UF1_semfile))
== (HANDLE) (NULL))

```

```

{
    /*
    ** if cannot open the semaphore, wait for 0.1
    second
    */
    fprintf(stderr, "Retry Open semaphore %s\n",
UF1_semfile);

    sleep(1);
}

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

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

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

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

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

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

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

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

get_time_stamp(T_STAMP_FORM_1,(Timer_struct *)NULL)); /* TIME_ACC jen*/

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

    if (!strcmp(userid,"0")) /* No authentication
provided */
        EXEC SQL CONNECT TO :UF_dbname;
    else

```

```

EXEC SQL CONNECT TO :UF_dbname USER
:UF_userid USING :UF_passwd;
error_check();

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

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

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

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

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

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

```

```

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

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

/* delete the orders now */

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

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

if (sqlcode == SQL_RC_E911)
{
/* we've hit a deadlock */

```

```

#ifdef UF2DEBUG
    fprintf (myostream, "orders deadlocked\n");
    fflush(myostream);
#endif
    fprintf (myostream,
        "\nA deadlock detected while deleting from
ORDERS: update pair %d set %d chunk %d.
Retrying.\n",
        updatePair, thisConcurrentDelete,
UF_chunk);
    dumpCa(&sqlca);
    SleepSome(UF_DEADLOCK_SLEEP);
    maxwait++; /* jen DEADLOCK */
}
else if (sqlcode < 0)
{
#ifdef UF2DEBUG
    fprintf (myostream, "orders failed\n");
    fflush(myostream);
#endif
    fprintf (myostream, "\nAn error %d
occurred deleting from
TPCD.ORDERS\n", sqlca.sqlcode);
    dumpCa(&sqlca);
    fprintf (myostream, "for update pair
number %d set %d chunk %d..Exiting\n",
        updatePair,
thisConcurrentDelete, UF_chunk);
    rc=-1;
}
else
{
#ifdef UF2DEBUG
    fprintf (myostream, "orders succeeded\n");
    fflush(myostream);
#endif
        /* accumulate the number of row deleted */
        /* Order count ONLY updated if both
orders and lineitem */
        /* go through */
        deletedLineitems +=
savedDeletedLineitems; /* kbs */
        deletedOrders += sqlca.sqlerrd[2];
        rc=0;
        EXEC SQL COMMIT WORK;
        error_check();
#ifdef UF2DEBUG
        /* report the number of rows deleted */
        fprintf(myostream, " interim %ld rows
for chunk %d from TPCD.ORDERS at %*.s\n",
            deletedOrders, UF_chunk, T_STAMP_1LEN, T_STAMP_1LEN, /* TIME_ACC jen*/
            T_STAMP_1LEN, T_STAMP_1LEN, /*
TIME_ACC jen*/
            get_time_stamp(T_STAMP_FORM_1, (Timer_struct *)NULL)); /* TIME_ACC jen*/
        deletedOrders, UF_chunk, T_STAMP_1LEN, T_STAMP_1LEN, /* TIME_ACC jen*/
            get_time_stamp(T_STAMP_FORM_1, (Timer_struct *)NULL)); /* TIME_ACC jen*/
        if (sqlca.sqlcode < 0)
        {
            fprintf (myostream, "# of deadlocks %d
exceeds %i\n", maxwait, MAXWAIT);
            rc=-1;
            EXEC SQL ROLLBACK WORK;
            error_check(); /* @d22275 tjt */
        }
        fprintf(myostream, " interim %ld rows
for chunk %d from TPCD.LINEITEM at %*.s\n",
            deletedLineitems, UF_chunk, T_STAMP_1LEN, T_STAMP_1LEN, /* TIME_ACC jen*/
            T_STAMP_1LEN, T_STAMP_1LEN, /*
TIME_ACC jen*/
            get_time_stamp(T_STAMP_FORM_1, (Timer_struct *)NULL)); /* TIME_ACC jen*/
        fprintf( myostream,
            " deletes for update pair %d chunk
%d complete at %*.s\n\n",
            updatePair, UF_chunk,
            T_STAMP_1LEN, T_STAMP_1LEN, /*
TIME_ACC jen*/
            get_time_stamp(T_STAMP_FORM_1,
                (Timer_struct *)NULL)); /*
TIME_ACC jen*/
        #endif
    } /* process orders deletes */
} /* while trying to delete one chunk loop */
} /* while there are more chunks */

#ifdef UF2DEBUG
    fprintf (myostream, "after loop\n");
    fflush(myostream);
#endif
    /* report the number of row deleted */
    fprintf(myostream, "%ld rows deleted from
TPCD.ORDERS at %*.s\n",
        deletedOrders, T_STAMP_1LEN, T_STAMP_1LEN,
/* TIME_ACC jen*/
        get_time_stamp(T_STAMP_FORM_1, (Timer_struct *)NULL)); /* TIME_ACC jen*/
    fprintf(myostream, "%ld rows deleted from
TPCD.LINEITEM at %*.s\n",
        deletedLineitems, T_STAMP_1LEN, T_STAMP_1LEN, /* TIME_ACC jen*/
        T_STAMP_1LEN, T_STAMP_1LEN, /* TIME_ACC jen*/
        get_time_stamp(T_STAMP_FORM_1, (Timer_struct *)NULL)); /* TIME_ACC jen*/
    if (sqlca.sqlcode < 0)
    {
        fprintf (myostream, "# of deadlocks %d
exceeds %i\n", maxwait, MAXWAIT);
        rc=-1;
        EXEC SQL ROLLBACK WORK;
        error_check(); /* @d22275 tjt */
    }
}

```

```

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

UF2_exit:
fclose (myostream);

/* exiting, increment the semaphore */
#ifdef SQLWINT
/* we used the tpcd.setup file to generate the
semaphore key begin SEMA */
if (getenv("TPCD_AUDIT_DIR") != NULL)
{
    sprintf(sourcefile, "%s%ctools%ctpcd.setup",
            getenv("TPCD_AUDIT_DIR"),
PATH_DELIM, PATH_DELIM);
}
else
{
    fprintf (stderr, "Can't open UF2 semaphore file
TPCD_AUDIT_DIR is not defined.\n");
    exit (-1);
}

    su_semkey = ftok (sourcefile, 'D'); /* use D for
deletes */
/* end SEMA */
while ((su_semid = semget(su_semkey,1,0)) < 0)
{
    if (errno == ENOENT)
        sleep(2);
    else {
        fprintf(stderr,"UF2 update stream %d: semget
failed errno = %d\n",
            updatePair, errno);
        exit(1);
    }
}
if (sem_op (su_semid, 0, 1) != 0) /*jenSEM*/
{
    /*jenSEM*/
    fprintf(stderr, "Failure to increment semaphore
UF2 set %d\n", thisConcurrentDelete);
    exit(1);
}
/*jenSEM*/

#else
    sprintf (UF2_semaphore, "%s.%s.UF2.semfile",
            getenv("TPCD_DBNAME"),
getenv("USER"));
    fprintf(stderr,"UF2 semfile = %s\n",UF2_semaphore);
    while ((su_hSem =
OpenSemaphore(SEMAPHORE_ALL_ACCESS |
SEMAPHORE_MODIFY_STATE |
SYNCHRONIZE,
TRUE,
UF2_semaphore))
== (HANDLE)(NULL)) {
/*
** if cannot open the semaphore, wait for 0.1
second
*/
        fprintf(stderr,"Retry Open semaphore %s\n",
UF2_semaphore);

        SleepSome(1);
    }

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

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

```


Appendix E: ACID Transaction Source Code

acid.h

```

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

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

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

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

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

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

struct update_struct {
    int qnum;
};

struct acidQ_struct {
    int tag;
    long o_key;

```

```

    double l_extendedprice;
};

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

/*
** in acid.sqc
*/

int updateQ (struct update_struct *us);

char del(void);

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

acid.sqc

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

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

```

```

*
*/

#include "acid.h"

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

#define DEADLOCK -911

/*
#define TRUNC2(d) ((floor((d)*100.0))/100.0)
*/
/*
#define TRUNC2(d)
((floor(nearest((d)*100.0)))*0.01)
*/
/*
#define TRUNC2(d)
((floor(nearest((d)*1000.0)/10.0)/100.0)
*/
/*
#define TRUNC2(d)
((floor(nearest((d)*10000.0)/1000.0)/100.0)
*/

void sqlerror(char * , struct sqlca *);

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

#ifdef SQLWINT

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

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

```

```

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

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

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

okey = acid->o_key;

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

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

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

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

/*
** use the same sql code as used in the
consistsql.pl to

```

```

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

EXEC SQL
SELECT

INTEGER(DECIMAL(SUM(DECIMAL(INTEGER
(INTEGER(DECIMAL

(INTEGER(100*DECIMAL(L_EXTENDEDPRICE
,20,3)), 20,3) *
(1-L_DISCOUNT)) *
(1+L_TAX)),20,3)/100.0),20,3) * 1000)
into :lEprice
FROM
TPCD.LINEITEM
WHERE
L_ORDERKEY = :okey;

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

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

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

rc = 0;
goto Qexit;

```

```

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

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

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

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

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

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

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

```

```

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

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

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

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

rc = 0;
goto Qexit;

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

Qexit:
fprintf(out, "\n----- END of ast_acidQ tag: %d -
-----\n\n",mypid);

```

```

fflush(out);fclose(out);
return(rc);
}
/*-----
_*/
/* acidT */
/*-----
_*/
int acidT (struct acidT_struct *acid)
{
time_t timeT;
FILE *out;
char out_fn[50];
struct timeval tv;
struct timezone tz;
int mypid;
int rc = 0;

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

EXEC SQL DECLARE l_cursor CURSOR FOR
SELECT l_partkey, l_suppkey, l_quantity,
l_tax, l_discount,
l_extendedprice
FROM tpcd.lineitem
WHERE l_orderkey = :o_key
AND l_linenumber = :l_key
FOR UPDATE OF l_extendedprice,
l_quantity;

EXEC SQL DECLARE o_cursor CURSOR FOR
SELECT o_totalprice
FROM tpcd.orders
WHERE o_orderkey = :o_key
FOR UPDATE OF o_totalprice;

if (acid->termination < 0 || acid->termination > 3)
acid->termination = 0;
o_key = acid->o_key;
l_key = acid->l_key;
delta = acid->delta;

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

```

```

    sprintf(out_fn,
"%s%acidT.out.%d",getenv("TPCD_TMP_DIR"),d
el(),mypid);
    out=fopen(out_fn,"a");
    gettimeofday(&tv, &tz);
    time(&timeT);
    fprintf(out,"\n----- START of acidT tag: %d
-----\n\n",mypid);
    fprintf(out, "acidT tag: %d, begin transaction
time: (%us %06uu) %s",
        mypid, tv.tv_sec, tv.tv_usec,
ctime(&timeT));
    fprintf(out, "o_key: %d\tl_key: %d\tdelta:
%d\n", o_key, l_key, delta);
}
#ifdef DEBUG
    printf("o_key: %d\tl_key: %d\tdelta: %d\n",
o_key, l_key, delta);
#endif

retry_tran:

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

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

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

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

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

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

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

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

if (sqlca.sqlcode != 0) {
    if(sqlca.sqlcode == DEADLOCK) goto
retry_tran;
    rc = sqlca.sqlcode;
    if (acid->logging) {
        fprintf(out,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
    }
}

```

```

    } else {
        fprintf(stderr,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
    } /* endif */
    sqlerror("acidT: UPDATE l_cursor", &sqlca);
    goto Terror;
}

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

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

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

EXEC SQL OPEN o_cursor;
if (sqlca.sqlcode != 0) {
    if(sqlca.sqlcode == DEADLOCK) goto
retry_tran;
    rc = sqlca.sqlcode;
    if (acid->logging) {

```

```

        fprintf(out,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
    } else {
        fprintf(stderr,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
    } /* endif */
    sqlerror("acidT: OPEN o_cursor", &sqlca);
    goto Terror;
}

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

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

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

#ifdef DEBUG
{
    double zeroone= l_extendedprice * (1.0-
l_discount);
    double zeroonetimes= (l_extendedprice * (1.0-
l_discount))*100.0;
    double firstone = TRUNC2(l_extendedprice *
(1.0-l_discount));
    double notone= TRUNC2 ( l_extendedprice *
(1.0-l_discount)) * (1.0+l_tax);
    double secondone= TRUNC2( TRUNC2(
l_extendedprice * (1.0-l_discount) ) * (1.0+l_tax) );

```

```

printf("firstone= %f\n", firstone);
printf("zeroone= %f\n", zeroone);
printf("zeroonetimes= %f\n", zeroonetimes);
printf("notone= %f\n", notone);
printf("secondone= %f\n", secondone);
}
#endif
ototal = o_totalprice -
    TRUNC2( TRUNC2( l_extendedprice *
(1-l_discount) ) * (1+l_tax) );
new_ototal = TRUNC2( new_extprice * (1.0-
l_discount) );
new_ototal = TRUNC2( new_ototal * (1.0+l_tax)
);
new_ototal = ototal + new_ototal;

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

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

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

/*
** why is this code in here? we don't want to
** commit until the history table has been updated
as well

```

```

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

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

EXEC SQL INSERT INTO tpcd.history values
(:l_partkey, :l_suppkey, :o_key, :l_key, :delta,
CURRENT TIMESTAMP);
if (sqlca.sqlcode != 0) {
if (sqlca.sqlcode == DEADLOCK) goto
retry_tran;
rc = sqlca.sqlcode;
if (acid->logging) {
fprintf(out,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
} else {
fprintf(stderr,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
} /* endif */
sqlerror("acidT: INSERT INTO history",
&sqlca);
goto Terror;
}

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

```

```

    }

    /* sleep for 1 second for 80% of the transactions */
#ifdef SQLWINT
    if ( ((rand() % (100)) + 1) < 80 ) sleep(1);
#else
    if ( ((random() % (100)) + 1) < 80 ) sleep(1);
#endif

    switch (acid->termination) {
    case 1:
    {
        if (acid->logging)
        {
            gettimeofday(&tv, &tz);
            time(&timeT);
            fprintf(out,"acidT tag: %d, wait before
COMMIT: (%us %06uu) %s",
                mypid, tv.tv_sec, tv.tv_usec,
ctime(&timeT));
        }
    }
    sleep(60);
    case 0:
    if (acid->logging) {
        gettimeofday(&tv, &tz);
        time(&timeT);
        fprintf(out,"acidT tag: %d, immediately before
COMMIT: (%us %06uu) %s",
            mypid, tv.tv_sec, tv.tv_usec,
ctime(&timeT));
    }
    EXEC SQL CLOSE L_CURSOR;
    EXEC SQL CLOSE O_CURSOR;
    EXEC SQL COMMIT;
    if (sqlca.sqlcode != 0) {
        if(sqlca.sqlcode == DEADLOCK) goto
retry_tran;
        rc = sqlca.sqlcode;
        if (acid->logging) {
            fprintf(out,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
        } else {
            fprintf(stderr,"acidT **ERROR** sqlcode =
%d\n",sqlca.sqlcode);
        } /* endif */
        sqlerror("acidT: COMMIT", &sqlca);
        goto Terror;
    }
    if (acid->logging) {
        gettimeofday(&tv, &tz);
        time(&timeT);
        fprintf(out,"acidT tag: %d, after COMMIT:
(%us %06uu) %s",
            mypid, tv.tv_sec, tv.tv_usec,
ctime(&timeT));
    }

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

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

```



```

rc = 0;
goto Texit;

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

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

/*-----
-*/
/*    updateQ                                */
/*-----
-*/

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

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

    qnum = us->qnum;

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

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

```

```

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

    sqlca.sqlcode = 0;
    discount = 0.25;
    price = 5000.50;
    acctbal = 1000.00;
    availqty = 10;
    size = 5;

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

        switch (qnum)
        {
            case 1:
                {
                    EXEC SQL
                    UPDATE TPCD.LINEITEM set
                    L_DISCOUNT = L_DISCOUNT + :discount
                    WHERE L_ORDERKEY IN
                    (326,512,928,995);
                    if (sqlca.sqlcode != 0) {
                        rc = sqlca.sqlcode;
                        if (acid->logging)
                        {
                            fprintf(out,"update query number: %d, pass
%d, **ERROR** sqlcode = %d\n",
                                qnum, i, sqlca.sqlcode);
                        }
                    }
                    else
                    {
                        fprintf(stderr,"update query number: %d,
pass %d, **ERROR** sqlcode = %d\n",
                            qnum, i, sqlca.sqlcode);
                    }
                    sqlerror("update query number 1", &sqlca);
                    goto Uerror;
                }
                discount = discount * (-1);
                secs2sleep = 300;
                break;
            }
            case 2:
                {
                    EXEC SQL

```

```

UPDATE TPCD.SUPPLIER set
S_ACCTBAL = S_ACCTBAL + :acctbal
WHERE S_NAME in
('Supplier#000000647','Supplier#000000070','Supplier#000000802');
if (sqlca.sqlcode != 0) {
rc = sqlca.sqlcode;
if (acid->logging)
{
fprintf(out,"update query number: %d, pass
%d, **ERROR** sqlcode = %d\n",
qnum, i, sqlca.sqlcode);
}
else
{
fprintf(stderr,"update query number: %d,
pass %d, **ERROR** sqlcode = %d\n",
qnum, i, sqlca.sqlcode);
}
sqlerror("update query number 2", &sqlca);
goto Uerror;
}
acctbal = acctbal * (-1);
secs2sleep = 90;
break;
}
case 3:
{
EXEC SQL
UPDATE TPCD.LINEITEM set
L_DISCOUNT = L_DISCOUNT + :discount
WHERE L_ORDERKEY IN (260930,
402497, 457859, 509889, 58117,
538311, 588421, 416167,
97830, 90276);
if (sqlca.sqlcode != 0) {
rc = sqlca.sqlcode;
if (acid->logging)
{
fprintf(out,"update query number: %d, pass
%d, **ERROR** sqlcode = %d\n",
qnum, i, sqlca.sqlcode);
}
else
{
fprintf(stderr,"update query number: %d,
pass %d, **ERROR** sqlcode = %d\n",
qnum, i, sqlca.sqlcode);
}
sqlerror("update query number 3", &sqlca);
goto Uerror;
}
discount = discount * (-1);
secs2sleep = 300;
break;
}

```

```

case 4:
{
if (i == 1) {
EXEC SQL
UPDATE TPCD.ORDERS set
O_ORDERDATE = O_ORDERDATE - 6
MONTHS
WHERE O_ORDERKEY = 67461;
/* WHERE O_ORDERKEY IN
(22400,28515,34338,46596,67461,92644,98307);*/
} else {
EXEC SQL
UPDATE TPCD.ORDERS set
O_ORDERDATE = O_ORDERDATE + 6
MONTHS
WHERE O_ORDERKEY = 67461;
}
if (sqlca.sqlcode != 0) {
rc = sqlca.sqlcode;
if (acid->logging)
{
fprintf(out,"update query number: %d, pass
%d, **ERROR** sqlcode = %d\n",
qnum, i, sqlca.sqlcode);
}
else
{
fprintf(stderr,"update query number: %d,
pass %d, **ERROR** sqlcode = %d\n",
qnum, i, sqlca.sqlcode);
}
sqlerror("update query number 4", &sqlca);
goto Uerror;
}
secs2sleep = 300;
break;
}
case 5:
{
EXEC SQL
UPDATE TPCD.LINEITEM set
L_DISCOUNT = L_DISCOUNT + :discount
WHERE L_ORDERKEY IN
(70976,566279,152897,84226,232483);
if (sqlca.sqlcode != 0) {
rc = sqlca.sqlcode;
if (acid->logging)
{
fprintf(out,"update query number: %d, pass
%d, **ERROR** sqlcode = %d\n",
qnum, i, sqlca.sqlcode);
}
else
{
fprintf(stderr,"update query number: %d,
pass %d, **ERROR** sqlcode = %d\n",

```

```

        qnum, i, sqlca.sqlcode);
    }
    sqlerror("update query number 5", &sqlca);
    goto Uerror;
}
discount = discount * (-1);
secs2sleep = 300;
break;
}
case 6:
{
    EXEC SQL
        UPDATE TPCD.LINEITEM set
L_DISCOUNT = L_DISCOUNT + :discount
        WHERE L_ORDERKEY in
(33,131,161,195,229,230,231,323,353,356);
    if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
            fprintf(out,"update query number: %d, pass
%d, **ERROR** sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        else
        {
            fprintf(stderr,"update query number: %d,
pass %d, **ERROR** sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 6", &sqlca);
        goto Uerror;
    }
    discount = discount * (-1);
    secs2sleep = 300;
    break;
}
case 7:
{
    EXEC SQL
        UPDATE TPCD.LINEITEM set
L_DISCOUNT = L_DISCOUNT + :discount
        WHERE L_ORDERKEY IN
(562917,410659,16550,398401,157634,429920,454
11);
    if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
            fprintf(out,"update query number: %d, pass
%d, **ERROR** sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        else
        {

```

```

            fprintf(stderr,"update query number: %d,
pass %d, **ERROR** sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
    }
    sqlerror("update query number 7", &sqlca);
    goto Uerror;
}
discount = discount * (-1);
secs2sleep = 300;
break;
}
case 8:
{
    EXEC SQL
        UPDATE TPCD.LINEITEM set
L_DISCOUNT = L_DISCOUNT + :discount
        WHERE L_ORDERKEY IN
(129569,343591,270242,254983,98500,28963);
    if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
            fprintf(out,"update query number: %d, pass
%d, **ERROR** sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        else
        {
            fprintf(stderr,"update query number: %d,
pass %d, **ERROR** sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 8", &sqlca);
        goto Uerror;
    }
    discount = discount * (-1);
    secs2sleep = 300;
    break;
}
case 9:
{
    EXEC SQL
        UPDATE TPCD.LINEITEM set
L_DISCOUNT = L_DISCOUNT + :discount
        WHERE L_ORDERKEY IN
(113509,232997,246691,379233,448162,32134);
    if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
            fprintf(out,"update query number: %d, pass
%d, **ERROR** sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        else
        {

```

```

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

```

```

        fprintf(stderr,"update query number: %d,
pass %d, **ERROR** sqlcode = %d\n",
            qnum, i, sqlca.sqlcode);
    }
    sqlerror("update query number 11", &sqlca);
    goto Uerror;
}
availqty = availqty * (-1);
secs2sleep = 180;
break;
}
case 12:
{
    if (i == 1) {
        EXEC SQL
        UPDATE TPCD.LINEITEM set
        L_RECEIPTDATE = L_RECEIPTDATE - 3
        YEARS
        WHERE L_ORDERKEY IN
        (33,70,195,355,677,837,960,962,1028);
    } else {
        EXEC SQL
        UPDATE TPCD.LINEITEM set
        L_RECEIPTDATE = L_RECEIPTDATE + 3
        YEARS
        WHERE L_ORDERKEY IN
        (33,70,195,355,677,837,960,962,1028);
    }
    if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
            fprintf(out,"update query number: %d, pass
%d, **ERROR** sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        else
        {
            fprintf(stderr,"update query number: %d,
pass %d, **ERROR** sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
    }
    sqlerror("update query number 12", &sqlca);
    goto Uerror;
}
secs2sleep = 300;
break;
}
case 13:
{
    EXEC SQL
    UPDATE TPCD.LINEITEM set
    L_DISCOUNT = L_DISCOUNT + :discount
    WHERE L_ORDERKEY IN
    (263,9476,32355,34854,53445,56901);
    if (sqlca.sqlcode != 0) {

```

```

        rc = sqlca.sqlcode;
        if (acid->logging)
        {
            fprintf(out,"update query number: %d, pass
%d, **ERROR** sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        else
        {
            fprintf(stderr,"update query number: %d,
pass %d, **ERROR** sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 13", &sqlca);
        goto Uerror;
    }
    discount = discount * (-1);
    secs2sleep = 90;
    break;
}
case 14:
{
    EXEC SQL
        UPDATE TPCD.LINEITEM set
L_DISCOUNT = L_DISCOUNT + :discount
        WHERE L_ORDERKEY IN
(32,225,326,448,449,483,512);
    if (sqlca.sqlcode != 0) {
        rc = sqlca.sqlcode;
        if (acid->logging)
        {
            fprintf(out,"update query number: %d, pass
%d, **ERROR** sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        else
        {
            fprintf(stderr,"update query number: %d,
pass %d, **ERROR** sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 14", &sqlca);
        goto Uerror;
    }
    discount = discount * (-1);
    secs2sleep = 180;
    break;
}
case 15:
{
    EXEC SQL
        UPDATE TPCD.LINEITEM set
L_DISCOUNT = L_DISCOUNT + :discount
        WHERE L_ORDERKEY IN
(1,4,7,35,135,131300);
    if (sqlca.sqlcode != 0) {

```

```

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

```

```

        rc = sqlca.sqlcode;
        if (acid->logging)
        {
            fprintf(out,"update query number: %d, pass
%d, **ERROR** sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        else
        {
            fprintf(stderr,"update query number: %d,
pass %d, **ERROR** sqlcode = %d\n",
                qnum, i, sqlca.sqlcode);
        }
        sqlerror("update query number 17", &sqlca);
        goto Uerror;
    }
    price = price * (-1);
    secs2sleep = 90;
    break;
}
default:
{
    fprintf(out,"ERROR: Invalid query number
specified %d\n", qnum);
    rc = 1;
    goto Uexit;
}
}

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

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

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

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

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

rc = 0;
goto Uexit;

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

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

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

```

```

        strcpy (dbname, dbname_ptr);
    }

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

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

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

    char err_fn[256];

    int j,k;

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

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

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

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

        fprintf(err_fp,"\n");
        fflush(err_fp);fclose(err_fp);
    }
}

#ifdef SQLWINT
void sleep(int sec)
{
    Sleep(sec * 1000);
}
#endif

char del(void)
{
#ifdef SQLWINT
    return "\\";
#else
    return '/';
#endif
}

#ifdef defined(SQLPTX) || defined(SQLWINT) ||
defined(SQLSUN) || defined(Linux)
/* added for PTX as this one is not there in libm */
double nearest(double x)
{

```

```

double y, z;

y = x;
if (x < 0)
    y = -x;
z = y - (int)y;
if (z == 0.5) {
    if ((int)floor(y) % 2) {
        return((x < 0) ? -ceil(y) : ceil(y));
    } else {
        return((x < 0) ? -floor(y) : floor(y));
    }
} else if (z < 0.5)
    return((x < 0) ? -floor(y) : floor(y));
else
    return((x < 0) ? -ceil(y) : ceil(y));
}
#endif /* SQLPTX */
-

```

makefile

```

DBNAME =
    $(TPCD_QUAL_DBNAME)

INCLUDE = $(HOME)/sqllib/include

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

#LFLAGS = -lm -lcurses -ls -ll -ly -liconv -lbsd
CFLAGS = -I$(INCLUDE) -
Dpascal= -DLINT_ARGS \
-DSQLA_NOLINES -DLinux
# .. sun -DSQLA_NOLINES

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

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

CC = g++

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

TARGET = $(EXEC) tsec

```

```

.SUFFIXES: .o .c .sqc .bnd

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

all: $(TARGET)

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

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

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

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

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

```


Appendix F: Price Quotations

07-25-03 10:10 From-SuSE Inc

+510 628 3381

T-129 P.002/002 F-067

Quotation

SuSE Inc.
318 Harrison St.
Suite 301
Oakland, CA 94607
USA

Quote Number:
183

Quote Date:
Jul 25, 2003

Page:
1

Quoted to:
IBM Corporation - Endicott
Accounts Payable
P.O.Box 9005
Endicott, NY 13761
USA

Customer ID	Good Thru	Payment Terms	Sales Rep
CCIBM02	8/24/03	Net 45 Days	SEHOL01

Quantity	Item	Description	Unit Price	Extension
1.00	2119-3INT-8	SuSE Linux Enterprise Server 8 Opteron for AMD64, Includes Media Kit and 1 Year Maintenance Program for 8 CPUs.	2,585.00	2,585.00
4.00	2119-3-MFJ-8	SuSE Linux Enterprise Server 8 for AMD64 - 1 Year Maintenance Program for 8 CPUs SuSE, Inc. confirms that these are official list prices for the products listed.	2,456.00	9,824.00
			Subtotal	12,409.00
			Sales Tax	
			Freight	
			Total	12,409.00



Home | About CDW | Customer Support | View Cart | Log

SmartSearch™

Brands: Hardware | Software | Networking | Accessories | Services

800 839 4239

RESOURCES

- Order Status
- My Company
- My Account
- Account Team
- New Accounts
- Rebates
- Special Events
- CDW Outlet
- Technical Support
- E-New sletters
- Solutions Library

ONLINE HELP

PRINTABLE VERSION



Image not Available

Manufacturer



Product Links

- > Similar Products
- > Send to Associate

QLOGIC 133MHZ PCI-X 2GB DUAL POR

Product Information



Usually Ships:

CDW Part:

Mfg. Part:

UNSPSC:

Sales Rank:

Price:

ADD

RESOURCES

- Order Status
- My Company
- My Account
- Account Team
- New Accounts
- Rebates
- Special Events
- CDW Outlet
- Technical Support
- E-Newsletters
- Solutions Library
- Reference Guides

ONLINE HELP

PRINTABLE VERSION

SHOPPING CART

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

YOUR SHOPPING CART :

Quantity	Product	CDW	Usually Ships	Price	Ext. Price
<input type="text" value="8"/>	Cisco 1000BASE-T GBIC Module	372656	Same Day	\$299.09	\$2,392.72
<input type="text" value="1"/>	Cisco Catalyst 3508G XL Enterprise Edition	184837	Same Day	\$3,599.74	\$3,599.74
<input type="text" value="3"/>	Cisco SMARTnet Onsite 1 Year 24x7 4 Hour	183063	1-2 Weeks	\$407.68	\$1,223.04

Click to remove an item from your cart

Sub-Total \$7,215.50

QuickCart:

Shipping Calc:

Enter a **CDW part number** to quickly add it to your cart.

Enter a postal code to quickly estimate shipping cost.

* **Sample: CDW Part #**

Usually Ships: [Same Day](#)
CDW Part:
 Mfg. Part: XXXXXX-XXXXXX
 UNSPSC: XXXXXXXX