



TPC Benchmarkä H (Decision Support) 10000 GB Full Disclosure Report

NCR 5350

Using Teradata DBS V2R5.0

Submitted for Review: December 23, 2002

Updated: March 12, 2003

ã December 23, 2002 NCR
All Rights Reserved, except as noted below

NCR makes no warranty of any kind with regard to the information contained in this publication, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. NCR shall not be held liable for errors contained herein nor for any damages including loss of profits or other incidental or consequential damages in connection with the furnishing or use of this document.

NCR believes the information contained herein is correct and complete as of the date of publication. The pricing information is believed to accurately reflect the prices in effect for the components, products, and services listed, as of the date of publication, using the pricing methods as described herein. Actual prices to any particular customer are dependent on business agreements, legal contracts, sales and delivery channel arrangements, etc. The performance information is believed to accurately reflect the performance of the components, products, and services listed, as of the date of publication. Actual performance experienced by any particular customer will not necessarily be the same as claimed herein due to the myriad of technical variables such as system layout and configuration, hardware and/or software revision levels and/or change notices, operations parameterization, and background system activity. The content of this document is for informational purposes only.

NCR is a registered trademark and the NCR design is a registered service mark of the NCR Company.

TPC Benchmark™ is a trademark of the Transaction Processing Performance Council.
The Teradata logo and Ynet are trademarks, and Teradata is a registered trademark of Teradata Corporation.
Pentium™ is a trademark of the Intel Corporation.

Other incidental trademarks appearing in this document are the trademarks of their respective companies.

PERMISSION IS HEREBY GRANTED TO COPY THIS PUBLICATION IN ITS ENTIRETY.

ONLY COPYING RIGHTS ARE GRANTED; ALL OTHER RIGHTS, INCLUDING RIGHTS OF AUTHORSHIP, OWNERSHIP, CONTENTS, AND PUBLICATION, ARE RESERVED.



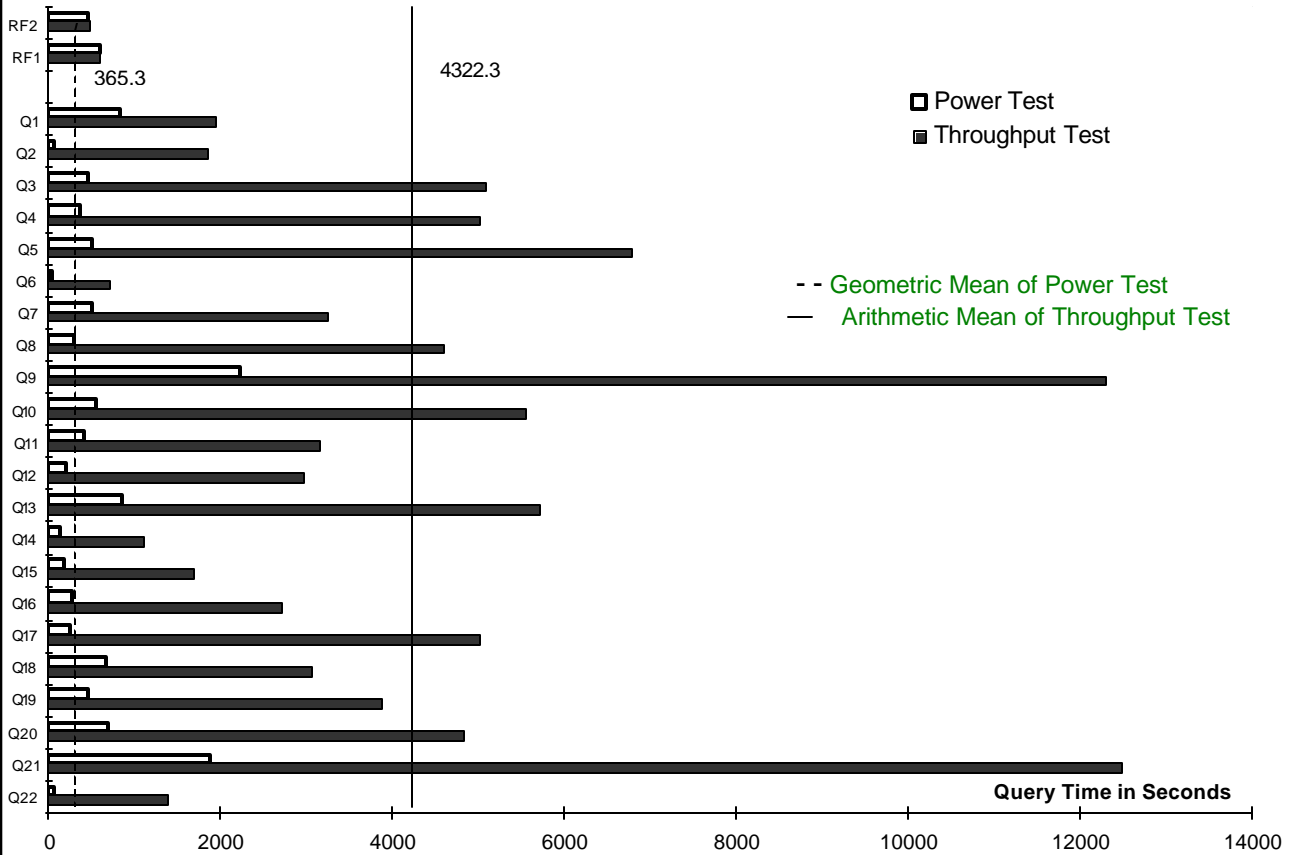
NCR 5350

Using Teradata V2R5.0

TPC-H Rev 2.0.0

Report Date: Dec. 23, 2002
Updated: Mar. 12, 2003

Total System Cost	Composite Query Per Hour Rating		Price/Performance	
\$ 19,774,904	81,501.6 QphH@10000GB		\$243 per QphH@10000GB	
Database Size	Database Manager	Operating System	Other Software	Availability Date
10000GB	Teradata V2R5.0	UNIX MP-RAS 3.02.00	None	HW: Now SW: Now



Database Load Time = 17Hrs 10 Mins Load included backup: N Total Data Storage / Database Size = 9.5

RAID (base tables): RAID (Base tables & Auxiliary Data Structures) = RAID (All): Y

System Configuration

Processors: 64 x 2 Intel Xeon 2.8 GHz CPUs w/512 KB Cache
 Memory (per node): 4 GB Memory
 Disk Drives (per **four** nodes): 3 Dual Controller Array each w/56 X 36GB 15K drives
 16 18GB internal drives (4 per node)
 BYNET interconnect support with 2 64-port BYNET switches

Total GB of Storage = 95409 GB

* Database size includes only raw data (e.g. no temp, index, redundant storage space, etc.)



NCR 5350 Teradata DBS V2R5.0

TPC-H Rev. 2.0.0
Report Date: Dec. 23, 2002
Updated: Mar. 12, 2003

PRODUCT ID	DESCRIPTION	QTY	EXT PRICE	3 - YR MAINT
Server Hardware				
9130-5354-8090	NCR 5350 QUAD NODE SYSTEM CABINET	1	\$455,000	\$44,100
9130-5358-8090	NCR 5350 QUAD NODE EXPANSION CABINET	15	\$6,840,000	\$661,941
9102-5301-8090	64 NODES BYNET V2 SWITCH CABINET, 5M CABLES BUNDLED	2	\$0	\$65,033
9130-F117	MEMORY DIMM SDRAM - 1GB	64	\$115,200	\$0
9130-F118	MEMORY DIMM SDRAM - 2GB	64	\$345,600	\$0
9130-F242	LSI QUAD FIBRE CHANNEL STORAGE ADAPTER, 2GB	128	\$153,408	\$0
2300-1602-8090	A16 AWS W2K 20+ NODES	1	\$8,700	\$772
2020-1021-8090	MONITOR 22IN	1	\$479	\$207
9130-F432	TAPE - DDS-4 SCSI 5GT 20/40GB	1	\$1,560	\$0
2300-K770	POWER CORD- US	1	\$21	\$0
2300-K800	KEYBOARD (US ENGLISH)	1	\$15	\$0
2020-K841	MODEM- EXTERNAL 56K USA	1	\$240	\$0
9130-K936	SIDE PANELS, STABILIZERS, MANUALS & ACCESSORIES	1	\$900	\$0
Server Subtotal			\$ 7,922,123	\$ 772,052
Storage Devices				
6000-9124-8090	WORLDMARK ENTERPRISE STORAGE CABINET	24	\$415,689	\$45,723
6840-1456-8900	HD QUAD MODULAR FC ARRAY 1456	48	\$1,766,400	\$275,184
6840-F206	DUAL FC CONTROLLER W/1GB CACHE	48	\$1,302,720	\$0
6840-F432	QA (4) 36GB 15K RPM DRIVES FC HD	672	\$3,249,469	\$219,300
6000-F262	INSTALL FEATURE 6840-1456	48	\$0	\$0
6000-K932	WES 5.0 ACCESSORIES KIT	3	\$0	\$0
6000-F951	12 METER FC CABLE WITH LC-LC	192	\$38,400	\$0
Storage Devices Subtotal			\$ 6,772,679	\$ 540,207
Software Licenses				
F601-7730-0000	SW; EOE/MPP 8 NODES	8	\$123,600	\$98,235
F601-7811-0000	SW; OLTP AWS OP ENVY	1	\$2,472	\$4,558
F7A4-1010-0000	TERADATA FASTLOAD FOR MPP UNIX NODE LICENSE	1	\$4,326	\$3,164
F7A4-0970-0000	TERADATA UTILITY PAK FOR MPP UNIX NODE LICENSE	1	\$4,326	\$836
F785-1762-0000	TERADATA/DB V2R5.0 MP-RAS 5350 0-10 TB FOR 1-90 NAMED USER	1	\$2,820,000	\$706,325
Software Subtotal			\$ 2,954,724	\$ 813,119
TOTAL			\$ 17,649,526	\$ 2,125,378
3-YEAR COST OF OWNERSHIP				\$ 19,774,904

"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 the stated prices are not available according to these terms, please inform the TPC at pricing@tpc.org. Thank you."

Audited by: Francois Raab
Infosizing (www.sizing.com)

Numerical Quantity Summary

Measurement Results:

Database Scale Factor	=	10000
Total Data Storage / Database Size	=	9.5
Start of Database Load Time	=	12-11-02 16:00:38
End of Database Load Time	=	12-12-02 09:10:08
Database Load Time	=	17 hrs. 10 min.
Query Streams for Throughput Test	=	9
TPC-H Power	=	98,551.1
TPC-H Throughput	=	67,401.7
Composite QphH@10000GB	=	81,501.6
Total System Price Over 3 Years	=	\$19,774,904
TPC-H Price/Performance Metric	=	\$243

Measurement Intervals:

Measurement Interval in Throughput Test (Ts) = 105,754 seconds

Duration of stream execution:

StreamID	Seed	Start Date	Start Time	End Date	End Time	Duration
Stream00	1212091008	12/14/02	03:09:17	12/14/02	06:44:52	03:35:35
Stream01	1212091009	12/14/02	06:44:52	12/15/02	09:24:29	26:39:36
Stream02	1212091010	12/14/02	06:44:52	12/15/02	09:17:33	26:32:40
Stream03	1212091011	12/14/02	06:44:52	12/15/02	09:08:23	26:23:31
Stream04	1212091012	12/14/02	06:44:52	12/15/02	08:24:32	25:39:39
Stream05	1212091013	12/14/02	06:44:52	12/15/02	09:26:21	26:41:28
Stream06	1212091014	12/14/02	06:44:52	12/15/02	09:07:05	26:22:12
Stream07	1212091015	12/14/02	06:44:52	12/15/02	08:58:19	26:13:26
Stream08	1212091016	12/14/02	06:44:52	12/15/02	09:18:18	26:33:25
Stream09	1212091017	12/14/02	06:44:52	12/15/02	09:22:32	26:37:39
Refresh		12/14/02	06:44:52	12/15/02	12:07:26	29:22:33

TPC-H Timing Intervals

Query	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
Stream0	838.5	72.1	454.5	373.7	504.2	47.9	511.1	274.9
Stream01	1953.1	2421.8	5142.1	1856.6	6879.9	656.4	3423.3	4443.9
Stream02	1759.4	2050.8	5251.1	4753.8	7261.0	1947.8	3030.1	3983.5
Stream03	1927.4	2078.3	4503.3	4980.7	6768.6	369.9	3178.1	5438.5
Stream04	1878.9	941.8	5046.7	5712.4	7485.6	572.6	4018.6	5334.8
Stream05	2697.7	1852.9	5208.9	3645.2	7372.1	258.7	1871.8	4584.5
Stream06	1799.1	1402.7	4741.8	6099.4	7259.2	412.9	3323.9	2566.5
Stream07	2085.7	2269.7	4567.2	6195.0	7858.9	603.9	3651.5	4145.0
Stream08	1440.1	1719.0	5332.3	6406.2	5404.4	954.3	2603.2	4948.8
Stream09	1802.7	1894.6	5935.6	5449.2	4918.8	565.8	4118.4	5968.2
Min Qi	1440.1	941.8	4503.3	1856.6	4918.8	258.7	1871.8	2566.5
Max Qi	2697.7	2421.8	5935.6	6406.2	7858.9	1947.8	4118.4	5968.2
Avg Qi	1927.1	1848.0	5081.0	5010.9	6800.9	704.7	3246.5	4601.5
Query	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16
Stream0	2217.3	554.6	423.5	196.2	857.6	127.2	185.2	273.6
Stream01	15987.2	5082.4	3943.8	3422.0	5291.5	1618.4	1428.3	3349.5
Stream02	12219.6	4338.3	4475.0	3669.8	6325.6	1208.6	1129.0	2826.4
Stream03	10212.3	7260.4	2931.2	2758.7	6780.0	928.9	1499.1	3420.7
Stream04	10768.5	5162.8	3396.8	2953.5	4621.6	705.1	1261.7	2931.5
Stream05	12547.9	5817.8	3065.0	3308.7	5183.7	1101.1	1560.3	2424.9
Stream06	14598.5	5369.8	2749.1	2637.0	5469.4	1203.7	1193.7	2684.2
Stream07	10394.5	5515.5	2676.8	2076.5	5778.9	1080.4	1881.8	3600.9
Stream08	12102.5	5116.0	1629.0	3152.8	6272.1	823.7	3275.9	2481.1
Stream09	11938.8	6468.9	3655.4	2820.9	5613.5	1283.0	2001.9	552.6
Min Qi	10212.3	4338.3	1629.0	2076.5	4621.6	705.1	1129.0	552.6
Max Qi	15987.2	7260.4	4475.0	3669.8	6780.0	1618.4	3275.9	3600.9
Avg Qi	12307.7	5570.2	3169.1	2977.8	5704.0	1105.9	1692.4	2696.9
Query	Q17	Q18	Q19	Q20	Q21	Q22	RF1	RF2
Stream0	247.0	655.1	451.9	677.7	1870.5	63.5	597.5	460.5
Stream01	5232.2	2776.1	3450.5	3662.2	12541.9	1413.6	590.9	469.5
Stream02	5267.2	3007.6	3281.8	5920.1	10426.2	1428.1	603.2	471.8
Stream03	3162.3	2943.0	3473.8	5875.9	13169.5	1350.6	606.6	477.8
Stream04	6031.4	2847.3	3205.1	4925.2	11258.0	1320.0	601.5	472.6
Stream05	6534.5	3250.7	4135.1	4912.0	13169.9	1585.4	603.1	472.6
Stream06	4789.3	3427.2	3950.7	4371.5	13023.8	1859.1	603.5	471.4
Stream07	5397.7	3985.8	4318.7	4390.0	10681.2	1251.1	596.6	473.4
Stream08	4987.4	2601.5	5354.0	4846.2	13152.8	1002.3	603.4	473.3
Stream09	3650.9	2709.0	3617.8	4616.3	15020.5	1257.3	601.4	471.9
Min Qi	3162.3	2601.5	3205.1	3662.2	10426.2	1002.3	590.9	469.5
Max Qi	6534.5	3985.8	5354.0	5920.1	15020.5	1859.1	606.6	477.8
Avg Qi	5005.9	3060.9	3865.3	4835.5	12493.8	1385.3	601.1	472.7



Benchmark Sponsor: Alain Crolotte
 NCR
 100 N. Sepulveda Blvd.
 El Segundo, CA 90245

December 17, 2002

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

Platform: **NCR 5350**
 Database Manager: **Teradata V2R5.0**
 Operating System: **UNIX MP-RAS 03.02.00**

The results were:

CPU (Speed)	Memory	Disks	QphH@10000GB
NCR 5350 (64 nodes, each with)			
2 x Intel Pentium IV (2.8 GHz)	512 KB Cache/cpu 4 GB Main	42 x 36 GB ext. 4 x 18 GB int.	81,501.6

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

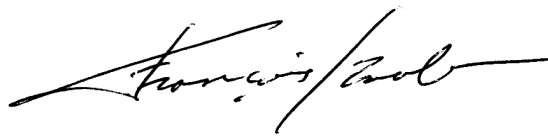
- The database records were defined with the proper layout and size
- The database population was generated using DBGEN
- The database was properly scaled to 10,000GB and populated accordingly
- The compliance of the database auxiliary data structures was verified
- The database load time was correctly measured and reported

- The required ACID properties were verified and met
- The query input variables were generated by QGEN
- The query text was produced using minor modifications
- The execution of the queries against the SF1 database produced compliant answers
- A compliant implementation specific layer was used to drive the tests
- The throughput tests involved 9 query streams
- The ratio between the longest and the shortest query was such that no query timing was adjusted
- The execution times for queries and refresh functions were correctly measured and reported
- The repeatability of the measured results was verified
- The required amount of database log was configured
- The system pricing was verified for major components and maintenance
- The major pages from the FDR were verified for accuracy

Additional Audit Notes:

None.

Respectfully Yours,

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

François Raab
President

TPC Benchmark H Overview

The TPC Benchmark™ H (TPC-H) is a Decision Support benchmark. It is a suite of business oriented queries and concurrent updates. The queries and the data populating the database have been chosen to have broad industry-wide relevance while maintaining a sufficient degree of ease of implementation. This benchmark illustrates Decision Support systems that

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

TPC-H evaluates the performance of various Decision Support systems by the execution of sets of queries against a standard database under controlled conditions. The TPC-H queries:

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

Table of Contents

PREFACE	XI
TPC Benchmark H Overview	xi
1. GENERAL ITEMS	1
1.1 Benchmark Sponsor(s)	1
1.2 Parameter Settings	1
1.3 Configuration Diagram	2
2. CLAUSE 1 LOGICAL DATABASE DESIGN	4
2.1 Database Definition Statements	4
2.2 Physical Organization	4
2.3 Horizontal Partitioning	4
2.4 Replication	3
3. CLAUSE 2 QUERIES AND REFRESH FUNCTIONS	5
3.1 Teradata SQL Used	5
3.2 Verifying Method for Random Number Generation	5
3.3 Generating Values for Substitution Parameters	5
3.4 Query Text and Output Data from Qualification Database	5
3.5 Query Substitution Parameters and Seeds Used	6
3.6 Isolation Level	5
3.7 Source Code of Refresh Functions	6
4. CLAUSE 3 DATABASE SYSTEM PROPERTIES	7
4.1 Atomicity	7
4.2. Consistency	7
4.3 Isolation	8
4.4 Durability	9

5. CLAUSE 4 SCALING AND DATABASE POPULATION	10
5.1 Ending Cardinality of Tables	10
5.2 Distribution of tables and logs across media	10
5.3 Modifications to the DBGEN	11
5.4 Database Load Time	11
5.5 Database Load Mechanism Details and Illustration	11
5.6 Data Storage Ratio	13
6. CLAUSE 5 PERFORMANCE METRICS AND EXECUTION RULES	14
6.1 Steps in the Power Test	14
6.2 Timing Intervals for Each Query and Refresh Function	15
6.3 Number of Streams for the Throughput Test	15
6.4 Start and End Date/Times for Each Query Stream	15
6.5 Total Elapsed Time of the Measurement Interval	15
6.6 Refresh Function Start Date/Time and Finish Date/Time	15
6.7 Timing Intervals for Each Query and Each Refresh Function for Each Stream	15
6.8 Performance Metrics	16
6.9 The Performance Metric and Numerical Quantities for Both Runs	16
6.10 Activity Between Runs	15
7. CLAUSE 6 SUT AND DRIVER IMPLEMENTATION	17
7.1 Description of Driver Performance Functions	17
7.2 Detailed Description of the Internal Driver Program and Implementation Layer Used	17
8. CLAUSE 7 PRICING	18
8.1 Detailed List of Hardware and Software Used	18
8.2 Total Three Year Price	18
8.3 Committed Delivery Date of General Availability	18
9. CLAUSE 9 AUDIT	19

9.1 Auditor's Information and Attestation Letter	19
APPENDIX A: OPERATING SYSTEMS AND DATABASE PARAMETERS AND OPTIONS	20
A.1 Operating System Parameters	20
A.2 Database Parameters	25
APPENDIX B. DATABASE DEFINITION STATEMENTS	26
B.1 Create Table Statements	26
B.2 Create Index and Collect Statistics Statements	27
APPENDIX C. QUERY TEXT AND OUTPUT DATA FROM QUALIFICATION DATABASE	29
APPENDIX D. QUERY SUBSTITUTION PARAMETERS AND SEEDS USED	63
APPENDIX E. DATA LOAD COMPONENTS	66
Description of the loading method	66
E.1 Inmod for FastLoad used by Pfast	66
E.2 Source Code for Pipeline-to-FastLoad Approach	90
E.3 Load Scripts	107
APPENDIX F. DRIVER AND IMPLEMENTATION SPECIFIC LAYER DETAIL	107
Driver program	130
Implementation specific layer	177

1. General Items

1.1 Benchmark Sponsor(s)

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

NCR is the sponsor of this TPC-H benchmark.

1.2 Parameter Settings

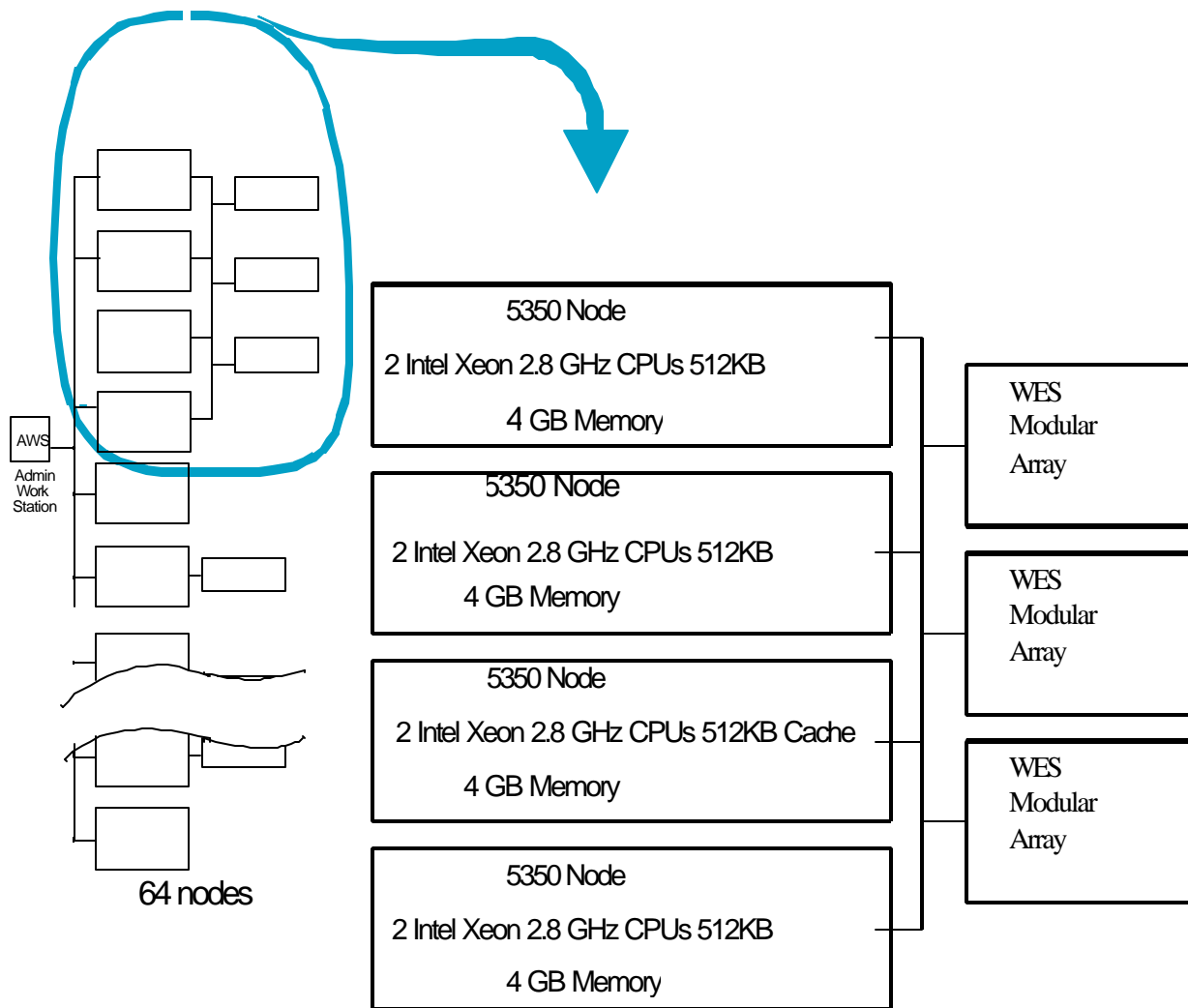
Settings must be provided for all customer-tunable parameters and options which have been changed from the defaults found in actual products, including but not limited to:

- Database tuning options*
- Optimizer/Query execution options*
- Query processing tool/language configuration parameters*
- Recovery/commit options*
- Consistency/locking options*
- Operating system and configuration parameters*

This requirement can be satisfied by providing a full list of all parameters and options.

See Appendix A for all Teradata Version 2 parameters.

1.3 Configuration Diagram



1.3.1 Measured and Priced Configuration Diagram

Server: 64 5350 Nodes each **four** nodes with:
 8 Intel Xeon 2.8Ghz CPUs w/512KB Cache
 16 GB Memory
 3 Dual Controller arrays w/ 56 x 36GB 15K drives
 4 BYNET Interface Cards
 4 CD-ROM Unit, Floppy Drive & Tape Drive
 4 Network cards
 16 18GB internal drives

BYNET interconnect support with:
 2 64-port BYNET switches

2. Clause 1 Logical Database Design

2.1 Database Definition Statements

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

See Appendix B for the Create Table statements. The actual programs and scripts that create tables, build indexes, and define collect statistics, which create and populate the Teradata database for the 5350 TPC-H benchmark are found in Appendix E.3.

2.2 Physical Organization

The physical organization of tables and indices, within the test and qualification databases, must be disclosed.

Teradata systems automatically handle data and index entry placement, without any manual organization or setup activity required. During this benchmark, no intervention or special options to Teradata's natural data placement were utilized.

2.3 Horizontal Partitioning

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

Horizontal partitioning was used for the lineitem and order tables.

2.4 Replication

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

No replication beyond the creation of indices was used

3. Clause 2 Queries and Refresh Functions

3.1 Teradata SQL Used

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

Teradata SQL was the query language used to implement all queries.

3.2 Verifying Method for Random Number Generation

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

The supplied DBGEN and QGEN methods were used to verify random numbers for the 5350 TPC-H benchmark.

3.3 Generating Values for Substitution Parameters

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

Values for the substitution parameter were generated from a customized internal driver program that embeds the functionality of the QGEN utility. The source code for this program is disclosed in Appendix G.

QGEN Version 1.5.0 was used in this benchmark execution.

3.4 Query Text and Output Data from Qualification Database

The executable query text used for query validation must be disclosed along with the corresponding output data generated during the execution of the query text against the qualification database. The output data for the power and throughput tests must be made available electronically upon request.

Appendix C contains the query text and query output.

3.5 Query Substitution Parameters and Seeds Used

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

Appendix D contains the seed and query substitution parameters.

3.6 Isolation Level

The isolation level used to run the queries must be disclosed.

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

3.7 Source Code of Refresh Functions

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

A specially-written driver program was implemented to execute both the power and the throughput tests, including RF1 and RF2, the update functions. This program uses DBGEN functionality to produce data and primary keys for use in both single and multi-stream tests.

Appendix F contains the source code of this program.

4. Clause 3 Database System Properties

4.1 Atomicity

4.1.1 Completed Transaction

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

The ORDER table total price and the LINEITEM table extended price were accessed, as well as a row count for the HISTORY table. The ACID Transaction was performed against ORDER and LINEITEM rows, changes were made and verified to the LINEITEM and ORDER tables, and it was proven that the correct data was inserted into the HISTORY table. Verification was done using these formulas:

- $L_Extendedprice = L_Extendedprice + ((\delta) * (L_Extendedprice / L_Quantity))$
- $O_Totalprice = O_Totalprice + (O_Cost * (1 - L_Discount) * (1 + L_Tax))$

4.1.2 Aborted Transaction

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

A count of the number of HISTORY table rows was obtained. The ACID Transaction was performed against randomly-selected rows, substituting a ROLLBACK for the COMMIT of the transaction.

The ORDER total price and the LINEITEM extended price were verified to be the same as they were prior to executing the ACID Transaction. Further, it was verified that no HISTORY record was inserted.

4.2. Consistency

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

4.2.1 Consistency Test

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

The initial consistency of the ORDER and LINEITEM tables was established, by proving this condition: $O_Totalprice = \text{Sum}(L_Extendedprice * (1 - L_Discount) * (1 + L_Tax))$. The prescribed number of ACID Transactions were executed and the consistency of the involved ORDER and LINEITEM rows was re-verified.

4.3 Isolation

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

4.3.1 Read-Write Conflict with Commit

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

Following the guidelines in the TPC-H Specification document, clause 3.4.2.1, it was verified that the second transaction does not see the first transaction's updates, and does not complete, until the first transaction has committed.

4.3.2 Read-Write Conflict with Rollback

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

Following the guidelines in the TPC-H Specification document, clause 3.4.2.2, it was verified that the second transaction does not see the first transaction's updates, and that the second transaction does not complete until the first transaction has rolled back.

4.3.3 Write-Write Conflict with Commit

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

Following the guidelines in the TPC-H Specification document, clause 3.4.2.3, it was verified that the second transaction, which attempts to update the same data as the first transaction, waits until the first transaction completes, and receives data that reflects the changes made by the first transaction.

4.3.4 Write-Write Conflict with Rollback

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

Following the guidelines in the TPC-H Specification document, clause 3.4.2.4, it was verified that the second transaction, which attempts to update the same data as the first transaction, waits until the first transaction rolls back, and receives data that reflects the original state of the data, not the changes made by the first transaction.

4.3.5 Read and Write Against Different Tables Concurrently

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

Based on the TPC-H Specification document, clause 3.4.2.5, it was proven that a second transaction that updates tables in the database not referenced by an earlier-started read-only transaction will complete prior to the read-only transaction. Verification was made that the appropriate rows were updated and that the update transaction started after and completed before the read only transaction.

4.3.6 Updates Not Indefinitely Delayed by Reads on Same Table

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

Following TPC-H Specification document, clause 3.4.2.6, process 1 submitted Query 1 (a long-running query) for execution, followed by a second process that executed a short update transaction against the same table. A third process submitted a second long-running query against the same table while the original process 1 was still executing. Process 1's query completed first, followed by process 2's update transaction, and lastly process 3's query completed. It was verified that the correct rows were updated and that updates are executed in a timely manner.

4.4 Durability

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

4.4.1 Failure of a Durable Medium

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

A disk failure was caused in the midst of the required durability test, the system was restarted, and the durability success file and the HISTORY table were compared successfully. The process recorded in the TPC-H Specification Document, Clause 3.5.2 was adhered to during this test.

4.4.2 System Crash

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

A system interruption was caused in the midst of the required durability test, the system was restarted, and the durability success file and the HISTORY table were compared successfully. The process recorded in the TPC-H Specification Document, Clause 3.5.3 as adhered to during this test.

4.4.3 Memory Failure

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

A memory failure was caused in the midst of the required durability test, the system was restarted, and the durability success file and the HISTORY table were compared successfully. The process recorded in the TPC-H Specification Document, Clause 3.5.3 was adhered to during this test.

5. Clause 4 Scaling and Database Population

5.1 Ending 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.

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

5.2 Distribution of tables and logs across media

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

In this configuration of 64 nodes, every four-node set has:

- 3 Fiber Channel WES modular array
- 168 x 36GB disk drives

Each 5350 node has been configured with 21 virtual AMPs, totalling 1344 virtual AMPs in the SUT.

The Teradata database system enforces the following data placement rules:

- The rows of each table are randomly assigned to one of the 1344 virtual AMPs.
- Each virtual AMP controls approximately an equal portion of each tables' rows.
- Partitioning for lineitem and orders is performed within each AMP. No partitioning is used for other tables.
- There is no knowledge of which physical disk specific rows reside on.
- The transient journal, responsible for logging before images of data changes, is local to each virtual AMP, and is therefore dispersed across disks similar to user data.

For lineitem and orders, the technique used in this benchmark is that of Partitioned Primary Index, a new feature in Teradata V2R5.0. The database selects an AMP for each row based on the primary index and if partitioning is used, rows are organized by partition (7 partitions were used in this benchmark) within each AMP. If partitioning is not used rows are organized sequentially, ordered by hash. Whether partitioning is used or not, data is evenly dispersed across all ranks within all disk arrays. Consequently, the following distribution summary is offered.

Every four 5350 node are associated with 3 Fiber Channel WES Modular Array. For each of the 64 nodes, the disks are utilized in the identical manner. The chart below represents a single 5350 node's associated disks.

The following description of data distribution is for one single node, and can be assumed to be identical for all 64 nodes in the system unless otherwise specified.

Disk Drive Description of Content of One Node

Internal 1	Operating system, root
Internal 2	DBGEN generated datasets for inserts on the first 15 nodes only.
Disk Array Module	1.5625 % of LINEITEM, ORDER, CUSTOMER, PART, SUPPLIER, PARTSUPP, NATION, REGION, TRANSIENT JOURNAL

NOTE: As the Region and Nation table have only 5 and 25 rows respectively, some virtual AMPs will not be assigned rows from those smaller tables.

5.3 Mapping of Database Partitions

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

The database was not replicated.

5.4 RAID Implementation

Implementations may use some form of RAID. The RAID level used must be disclosed for each device. If RAID is used in an implementation, the logical intent of its use must be disclosed.

RAID 1 (mirroring) was used for everything.

5.5 Modifications to the DBGEN

Any modifications to the DBGEN (see Clause 4.2.1) source code (provided in Appendix B) 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.

Appendix E contains the source of the inmods for the PFast utility program. Section 5.7 of this FDR provides detail about this utility program. DBGEN was modified to allow scale factor 10,000 (C_NAME size increased by 1)

5.6 Database Load Time

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

The database load time was 17 hours 10 minutes.

5.7 Database Load Mechanism Details and Illustration

The details of the database load mechanism must be described and illustrated with a block diagram.

Two different methods were used to load the TPC-H database. Both incorporated the standard Teradata FastLoad utility, and both are described and illustrated below.

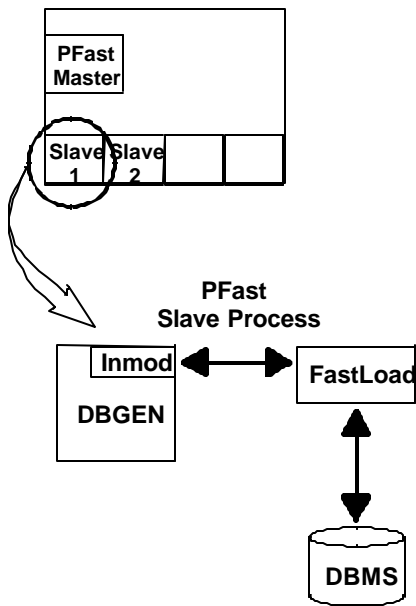
5.7.1 Parallel FastLoad

Parallel FastLoad (PFast) is a product that extends the functionality of the standard Teradata FastLoad utility. PFast, which extends parallelism to the client portion of FastLoad, has two components: A master process that executes on a single node along with four slave processes. The master process begins and coordinates the entire database load activity, while each of the slave processes do the actual generation (or reading) of rows and FastLoad execution. PFast loads data to a table through multiple parallel streams.

For this benchmark, PFast is used to load all tables: Lineitem, Order, Customer, Part, Supplier, Partsupp, Region, Nation. Two slave processes are activated on the SUT, each generating 1/2 of the rows feeding the database on the 64-node system.

The FastLoad utility is the driver within each slave process, calling DBGEN as additional rows are required. The generated rows then pass through an inmod (specific to the TPC-H database), and control is passed back to the FastLoad instance which loads the data. Each slave has a copy of the DBGEN code and the inmod.

Appendix E contains the inmods used by Pfast.



GD01B008

5.7.2 Pipeline-to-FastLoad

With the Pipeline-to-FastLoad approach, data was generated using the standard TPC-H data generator executing from the UNIX test platform. A customized procedure was designed, in which rows generated moved through a UNIX named pipe to a continuously active single instance of the Teradata FastLoad utility. This Teradata FastLoad utility loaded the table data in parallel across all virtual AMPs, and did not require any presorting or staging of the data.

The Nation and Region tables were loaded concurrently using this pipeline technique, so for some of this loading interval, two concurrent Fastload jobs were active at the same time.

Appendix E contains the source code for the Pipeline-to-FastLoad technique.



5.8 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 4.1.3.1.

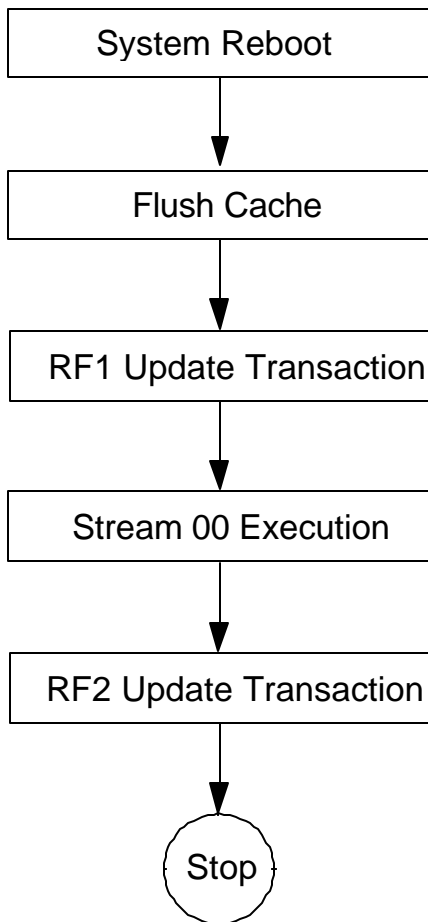
The data storage ratio is computed based on the following information:

Disk Type	# of Disks	Space per Disk	Sub-Total Disk Space	Total Size	Data Storage Ratio
External	2688	33.88 GB	91069GB		
Internal	256	16.92 GB	4332 GB	95401 GB	9.5

6. Clause 5 Performance Metrics and Execution Rules

6.1 Steps in the Power Test

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



GD01B002

6.2 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.

The Power Test timing intervals can be found in the Executive Summary which is placed at the beginning of this document, under the Numerical Quantity Summary.

6.3 Number of Streams for the Throughput Test

The number of execution streams used for the Throughput Test must be disclosed.

The Throughput Test used 9 execution streams.

6.4 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.

The Throughput Test start time and finish time for each stream is reported in the Executive Summary of this document within the Numerical Quantity Summary.

6.5 Total Elapsed Time of the Measurement Interval

The total elapsed time of the measurement interval:

The total elapsed time of the throughput test was 105,754 seconds.

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

StreamID	RF1 StartDate/Time		RF2 StartDate/Time	
	RF1 EndDate/Time		RF2 EndDate/Time	
Stream00	12/14/2002	3:09:17	12/14/2002	6:37:12
	12/14/2002	3:19:14	12/14/2002	6:44:52
Stream01	12/15/2002	9:26:21	12/15/2002	9:36:12
	12/15/2002	9:36:12	12/15/2002	9:44:02
Stream02	12/15/2002	9:44:02	12/15/2002	9:54:05
	12/15/2002	9:54:05	12/15/2002	10:01:57
Stream03	12/15/2002	10:01:57	12/15/2002	10:12:03
	12/15/2002	10:12:03	12/15/2002	10:20:01
Stream04	12/15/2002	10:20:01	12/15/2002	10:30:03
	12/15/2002	10:30:03	12/15/2002	10:37:55
Stream05	12/15/2002	10:37:55	12/15/2002	10:47:58
	12/15/2002	10:47:58	12/15/2002	10:55:51
Stream06	12/15/2002	10:55:51	12/15/2002	11:05:54
	12/15/2002	11:05:54	12/15/2002	11:13:46
Stream07	12/15/2002	11:13:46	12/15/2002	11:23:42
	12/15/2002	11:23:42	12/15/2002	11:31:36
Stream08	12/15/2002	11:31:36	12/15/2002	11:41:39
	12/15/2002	11:41:39	12/15/2002	11:49:32
Stream09	12/15/2002	11:49:32	12/15/2002	11:59:34
	12/15/2002	11:59:34	12/15/2002	12:07:26

6.7 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.

The timing intervals for each query and update function for the Throughput Test is reported in the Executive Summary at the beginning of this document, within the Numerical Quantities Summary.

6.8 Performance Metrics

Verify that the metrics are computed as required.

The performance metrics, and the numbers on which they are based, are contained within the Executive Summary at the beginning of this document, within the Numerical Quantities Summary.

6.9 The Performance Metric and Numerical Quantities from Both Runs

The performance metric (QphH@Size) and the numerical quantities (TPC-H Power@Size and TPC-H Throughput@Size) from both of the runs must be disclosed (see Clause 5.4.1)..

Two consecutive runs were executed. The following table contains the performance metric and numerical quantities from both runs.

System	QppH@10000GB	QthH@10000GB	QphH@10000GB
Run 1	99,355.6	68,466.0	82,477.1
Run 2	98,551.1	67,401.7	81,501.6

6.10 Activity Between Runs

Any activity on the SUT that takes place between the conclusion of Run 1 and the beginning of Run 2 must be fully disclosed including listings of scripts or command logs along with any system reboots or database restarts.

The system was restarted between the end of Run 1 and the beginning of Run 2.

7. Clause 6 SUT and Driver Implementation

7.1 Description of Driver Performance Functions

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

No driver external to the SUT was used.

7.2 Detailed Description of the Internal Driver Program and Implementation Layer Used

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 internal driver program used in this benchmark utilizes the standard Teradata call-level interface (CLIV2). An implementation specific layer was added between the driver program and CLIV2 to hide some of the complexities of CLIV2 from the main body of the program. This program provides the following functionality:

- The program reads the same input files as QGEN and generates the query text with the appropriate input variables.
- The program directly submits the queries, with a time-date stamp between each query submission.
- The program produces two output files: 1) A summary output that includes the query times, and 2) Detail output which includes all of the query text and output.

Source code of the Driver Program and Implementation Specific Layer is included in Appendix F.

7.3 Profile-Directed Optimization

If profile-directed optimization as described in Clause 5.2.9 is used, such use must be disclosed. In particular, the procedure and any scripts used to perform the optimization must be disclosed.

Profile-directed optimization was not used.

8. Clause 7 Pricing

8.1 Detailed List of Hardware and Software Used

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

A detailed list of hardware and software used in this benchmark, as well as the total three-year price calculations, is listed in the Executive Summary at the beginning of this document. This pricing information was verified with the NCR Customer Support Services in Dayton, Ohio.

8.2 Total Three Year Price

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

NCR's Standard Pricing methodology has been used in the pricing of this configuration. The hardware service scope includes one hour remote problem acknowledgement and the presence on site within 4 hours of either a customer replaceable part or a qualified maintenance engineer. The software service scope includes four hour acknowledgement of software problems via the Web. See the Pricing Summary at the beginning of this document

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

Mr. Jeffrey Shoemacher, Director of Pricing

email: Jeffrey.Shoemacher@ncr.com phone: (937)445-7865

8.3 Committed Delivery Date of General Availability

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

Teradata V2R5.0 software is available now.

5350 hardware is available now.

9. Clause 9 Audit

9.1 Auditor's Information and Attestation Letter

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

An attestation letter is attached to the summary report, as well as the auditor's name and address. This letter was submitted at the time of benchmark publication.

Appendix A: Operating Systems and Database Parameters and Options

A.1 Operating System Parameters

```
*****
Unix mtune file
*****
* Lines ending in "%%INS%" are from mtune.d/* %%INS%%
* and constructed automatically. %%INS%%
* DO NOT edit manually. %%INS%%
YNET 1 1 1 %%INS%%
NFS_WSYNC 0 0 1 %%INS%%
NFS3_WSYNC 0 0 1 %%INS%%
* The following lines, if any, are entries %%INS%%
* preserved from the previous mtune file. %%INS%%
* Copyright 1994. 1996 AT&T Global Information Solutions -
Dayton, Ohio, USA
*
* Copyright (c) 1990 UNIX System Laboratories, Inc.
* Copyright (c) 1984, 1986, 1987, 1988, 1989, 1990 AT&T
* All Rights Reserved
*
* THIS IS UNPUBLISHED PROPRIETARY SOURCE CODE OF
* UNIX System Laboratories, Inc.
* The copyright notice above does not evidence any
* actual or intended publication of such source code.
*
* Copyright (c) 1987, 1988 Microsoft Corporation
* All Rights Reserved
```

```
*
* This Module contains Proprietary Information of Microsoft
* Corporation and should be treated as Confidential.
*
*ident "@(#) :mtune 23.1.2.3"
*
* Modifications & additions have been made to the mtune file for
performance
* tuning. Consult tuning documents for explanations.
* Modified: BUFHWM SYSSEGSZ SEGMAPSZ GPGSLO GPGSHI
* Added: LOTSFREE DESFREE MINFREE
* Deleted: SYSSEGSZ
*
* "OBSOLETE" in second field means parameter is no longer used by
the OS.
* The parameter may be removed in a later version of the OS.
Idconfig
* will a print warning if the parameter is used in stune.
*
* General Kernel Parameters -----
TRACESZ 8192 4096 8192
NCPUR 32 1 32
NCALL 60 30 700
NFILE OBSOLETE
NMOUNT OBSOLETE
NPROC 250 50 10240
NREGION OBSOLETE
NCLIST 120 1 400
MAXUP 30 15 2048
NOFILES OBSOLETE
NHBUF 64 32 1024
NPBUF 20 20 8192
NAUTOUP 60 0 120
FDFLUSHR 1 1 1
BDFLUSHR OBSOLETE
MAXPMEM 0 0 16384
SHLBMAX 3 2 6
FLCKREC 300 100 2000
PUTBUFSZ 5120 2000 20000
MAXSLICE 100 25 100
ULIMIT OBSOLETE
SPTMAP 400 50 600
PIOMAP 50 50 50
PIOMAXSZ 64 4 64
MAXMINOR 0x3ffffff 255 0x3ffffff
NGROUPS_MAX 16 1 16
NBUF 100 100 3000
```

```

BUFHWM      0      0      65536
ARG_MAX     5120   1024   51200
RSTCHOWN    0      0      1
MAXLINK     2048   1000   32767
* File System Parameters -----
NINODE      500    100    5000
NS5INODE    OBSOLETE
UFSNINODE   500    100    5000
VX_NINODE   0      0      10000
NDQUOT      200    100    400
NRNODE      300    100    1300
S52KNBUF    OBSOLETE
S52KNHBUF    OBSOLETE
* ----- CDFS Parameters -----
NCDINODE    256    64     2048
NCDEXTENT   OBSOLETE
NCDFILSYS   OBSOLETE
* Paging Parameters -----
VHNDFRAC    OBSOLETE
AGEINTERVAL OBSOLETE
MINPAGEFREE 64     64     64
GPGSLO      25     0      25
GPGSHI      OBSOLETE
GPGSMSK     OBSOLETE
MAXSC        OBSOLETE
MAXFC        OBSOLETE
MAXUMEM     OBSOLETE
MINARMEM    100    100    160
MINASMEM    25     25     40
MINAKMEM    16     4      64
MINHIDUSTK  4      4      32
MINUSTKGAP  2      2      32
PAGES_UNLOCK 200    200    200
* Pageout Daemon Parameters
LOTSFREE    0      0      512
DESFREE     0      0      256
MINFREE     0      0      128
* STREAMS Parameters -----
NQUEUE      OBSOLETE
NSTREAM     OBSOLETE
NSTRPUSH    9      9      10
NSTRREVENT  OBSOLETE
MAXSEPGCNT  OBSOLETE
NMUXLINK    OBSOLETE
STRMSGSZ    0      0      32767
STRCTLSZ    1024   1024   1024
STRTHRESH   0x400000 0      0x2800000

```

```

NBLK4096    OBSOLETE
NBLK2048    OBSOLETE
NBLK1024    OBSOLETE
NBLK512     OBSOLETE
NBLK256     OBSOLETE
NBLK128     OBSOLETE
NBLK64      OBSOLETE
NBLK16      OBSOLETE
NBLK4       OBSOLETE
STRLOFRAC   OBSOLETE
STRMEDFRAC  OBSOLETE
* ----- muoe952386: Made FASTBUF tunable -----
* If the value is set to 0, then the system will determine the
default value
* for this variable based on system size.
*
* NOTE: If FASTBUF value was changed from 0 the new value should
be 128 atleast
* and multiple of four bytes.
* -----
FASTBUF      0      0      512
* ----- log (strlog) related tunable parameters
NLOG         10     3      16
MAXWERRMSG   80     0      100
MAXWTRCMSG   80     0      100
MAXWCONSMMSG 80     0      100
NUMSP        64     5      128
NUMTIM       128    1      8192
NUMTRW       16     1      8192
TIM_HIWATER  OBSOLETE
TIM_LOWATER  OBSOLETE
TRW_HIWATER  65535  512    1048576
TRW_LOWATER  40960  512    1048576
SOCK_HIWATER 8192   512    1048576
SOCK_LOWATER 4096   128    1048576
NUMSAD       8      1      16
NSTRPHASH    64     16     512
NAUTOPUSH    32     32     32
* Message Parameters -----
MSGMAP       100    10     1600
MSGMAX       2048   512    32767
MSGMNB       4096   4096   65535
MSGMNI       50     50     1000
MSGSSZ       8      8      8
MSGTQL       40     40     512

```

MSGSEG 1024 1024 65535

* Semaphore Parameters -----

SEMMAP 25 10 1000
SEMMNI 25 10 4096
SEMMNS 60 60 4096
SEMMNU 30 30 10240
SEMMSL 25 25 1000
SEMOPM 10 10 32
SEMUME 10 10 32
SEMVMX 32767 32767 32767
SEMAEM 16384 16384 16384

* Shared Memory Parameters -----

SHMMAX 524288 131072 0x7FFFFFFF
SHMMIN 1 1 1
SHMMNI 100 100 500
SHMSEG 6 6 64
SHMALL OBSOLETE

* Shared Memory Nailing GIDs -----

SHM_NAILED_GID1 0 0 2147483647
SHM_NAILED_GID2 0 0 2147483647
SHM_NAILED_GID3 0 0 2147483647
SHM_NAILED_GID4 0 0 2147483647
SHM_NAILED_GID5 0 0 2147483647
SHM_NAILED_GID6 0 0 2147483647
SHM_NAILED_GID7 0 0 2147483647
SHM_NAILED_GID8 0 0 2147483647
SHM_NAILED_GID9 0 0 2147483647

* RFS Parameters -----

NRCVD 150 40 500
NSNDD 150 100 350
NSRMOUNT 20 1 50
NADVERTISE OBSOLETE
MAXGDP 24 10 32
MINSERVE 3 3 3
MAXSERVE 6 3 6
NRDUSER 250 1 700
RFHEAP OBSOLETE
NLOCAL OBSOLETE
NREMOTE OBSOLETE
RCACHETIME 10 -1 10
RFS_VHIGH OBSOLETE
RFS_VLOW OBSOLETE
RF_MAXKMEM 0 0 50

* XENIX Parameters -----

DSTFLAG 1 0 1
NSCRN 0 0 10
NEMAP 10 10 10

TIMEZONE 480 0 1440
XSEMMAX 60 0 60
XSDSEGS 25 0 100
XSDSLOTS 3 0 5

* Miscellaneous Parameters -----

DO386B1 2 0 2
DO387CR3 2 0 2
SANITYCLK 0 0 1
DMAEXCL 1 0 1
MAXDMAPAGE 0 0 524288
DMAABLEBUF 70 10 100
KDBSYMSIZE 200000 10000 400000
PIOSEGSZ 1024 1024 1024
SEGMAPSZ 0 0 51200
FORCESMALLMEMORY 0 0 1

* Device Driver Parameters -----

NUMXT 3 1 3
NUMSXT 6 1 6
NCPYRIGHT 10 10 10
NKDVTTY OBSOLETE
PRFMAX 4096 2048 10240
CMF 1 0 1
RCMF 0 0 1
COM2CONS 0 0 1
RIDEOUT OBSOLETE
MNR_ON OBSOLETE
SANECNT OBSOLETE
USANEON OBSOLETE

* ASYNCIO Parameters -----

NAIOSYS 300 0 4096
MINAIOS OBSOLETE
MAXAIOS 5 1 64
AIOTIMEOUT OBSOLETE
NAIOPROC OBSOLETE

* EVENTS Parameters -----

MEVQUEUES 50 50 50
MEVKEVS 50 50 50
MEVEXREFS 50 50 50
MEVEXPRS 50 50 50
MEVTERMS 250 250 250
MEVSEXPRES 50 50 50
MEVSTERMS 100 100 100
MEVTIDS 100 100 100
MEVRETRYS 50 50 50
MEVEXITS 50 50 50
MEVSIGS 50 50 50
MEVSTRDS 50 50 50

```

MEVDIRENTS    50    50    50
EVDDATA      20480  20480  20480
EVTIDHTS     128    128    128
EVFNHTS      256    256    256
EVMAXEV      20     20     20
EVMAXDPE     1024   1024   1024
EVMAXMEM     10240  10240  10240
EVMAXTRAPS   25     25     25
EVMAXETERMS  20     20     20
* Timer and Scheduler Parameters -----
HRTIME       50     50     50
HRVTIME      50     50     50
RTMAXPRI     OBSOLETE
RTNPROCS     OBSOLETE
TSMAXUPRI    20     10     30
TSNPROCS     OBSOLETE
MAXCLSYPRI   160    160    160
* Affinity Scheduling Tunables -----
AFFIN_ON     1      0      1
AFFINDECAY   16     0      512
* Resource Limit Parameters -----
*
* Default per process resource limits (set to 0x7FFFFFFF for
infinite limit)
* S prefix is for soft limits, H prefix is for hard limits
*
* CPULIM - maximum combined user and system time in seconds
* FSZLIM - maximum file size in bytes
* DATLIM - maximum writeable mapped memory (swap space) in bytes
* STKLIM - maximum size of current stack in bytes
* CORLIM - maximum size of core file in bytes
* FNOLIM - maximum number of file descriptors
* VMMLIM - maximum amount of simultaneously mapped virtual memory
in bytes
SCPULIM      0x7FFFFFFF  60    0x7FFFFFFF
HCPULIM      0x7FFFFFFF  60    0x7FFFFFFF
SFSZLIM      0x7FFFFFFF  0x100000  0x7FFFFFFF
HFSZLIM      0x7FFFFFFF  0x100000  0x7FFFFFFF
SDATLIM      0x1000000  0x1000000  0x7FFFFFFF
HDATLIM      0x1000000  0x1000000  0x7FFFFFFF
SSTKLIM      0x1000000  0x2000 0x7FFFFFFF
HSTKLIM      0x1000000  0x2000 0x7FFFFFFF
SCORLIM      0x2000000  0x100000  0x7FFFFFFF
HCORLIM      0x2000000  0x100000  0x7FFFFFFF
SFNOLIM      0x40    0x20  0x400
HFNOLIM      0x400  0x20  0x400
SVMLIM      0x1000000  0x1000000  0x7FFFFFFF

```

```

HVMMMLIM      0x1000000  0x1000000  0x7FFFFFFF
* Number of overflow buffers in pageio_setup
PGOVERFLOW    16     8     128
NOTPGOVERFLOW 16     8     128
* bdevcnt, cdevcnt tunable
BMAX          25     20     256
CMAX          40     30     256
NCR_3500      0      0      0
MULTI_PROC    0      0      0
* ----- Level 5 Tunable Parameters -----
L5SysInt_Type 5      4      5
L5NMInables  0xff    0      0xff
L5SysIntenables 0      0      0xff
L5CPUenables 0xef    0      0xff
L5McAddrenables 0xfd    0      0xff
L5McDataenables 0x87    0      0xff
L5DMAenables  0xc0    0      0xff
SBithreshold  50      0      0x7fffffff
SBIDisablePeriod 1440   0      0x7fffffff
L5WatchDogPeriod 15      0      0x7fffffff
L5WatchDogPFRPeriod 1800  0      0x7fffffff
L5FrontSwitchTime 0      -2     60
* ----- End of Level 5 Tunable Parameters -----
* ----- performance Group Tunables -----
JQF_CLIENTS   2500  100   3000
JQF_SIZE      40     40    300
JQF_SESS      10     10    50
PW_MAX_ORA_PID OBSOLETE
PW_MAX_EVENT  OBSOLETE
* muoe961889 : hbc - 27 Jun 1996
* Increased LISTIO_MAX_CNT to 1024
LISTIO_MAX_CNT 1024  1     2048
TUBESIZE       1024  1024  4096
TUBETYPE       OBSOLETE
* ----- End of Performance group Tunables -----
* SUM Privilege Module tunable -----
PRIVID        0      0      0
* ----- Override Autocad lockout mechanism -----
ACADVERRIDE    0      0      1
* tunables for loadable kernel modules
BDEV_RESERVE  10     0      255
CDEV_RESERVE  25     0      255
FMODE_RESERVE 15     0      255
VFS_RESERVE   10     0      255
DEF_UNLOAD_DELAY 600  0      3600
* End of tunables for loadable kernel modules
* Added thru MR muoe951326, 04/09/95

```

```

* OS_COMPAT --- kernel compatibility options ---
* 0 Use default behavior
* 1 Use traditional behavior
* 2 Use new SVR4.2 behaviors
OS_COMPAT      1      0      2
* ----- RPC tunables muoe952511, 7/12/95 */ -----
RPC_HIWATER    0      0      500000
RPC_LOWATER    0      0      500000
* Raw Device Async I/O parameters -----
MAXFREE_RAIO  128     32     512
MINFREE_RAIO  32      0      128
MAXRAIO        256     32     4096
* More Raw Device Async I/O parameters -----
RAIO_TOGGLE   1      0      1
* UnixWare-compatible async I/O parameters
NUMAIO        128     32     4096
NUMPROCAIO    32      4     1024
AIO_LISTIO_MAX 100     32     1024
AIO_MAX        1      1      1
AIO_LOCK_ON_DEMAND 0      0      1
VDMAXRETRY    1      0      2
NMBYTE        256     256    65536
KMEM_MINIPAGES 100     8      512
KMEM_DESIPAGES 100     8      512
DMEM_DESIPAGES 100     8      512
BK_SPTALLOC_SZ 0      100    500
* --- DA0EXTN additions for Tunable Parameters (Fri Mar 22
15:20:46 EST 2002) ---
VDASYNCMAX    50      0      100
VDASYNCPARITY 1      0      1
VDASYNCWITES  1      0      1
VDDELAY        0      0      10
VDDISTJOBS    500     100    10000
VDDISTOOS     0      0      1
VDHASHMAX     2048    512    8192
VDILOGFLUSH   600     30     65535
VDJOBS        1000    200    10000
VDKEEPALIVE   15      5      30
VDMPPGCPUCNT  1      1      32
VDPREEMPTCNT  10      1      20
VDRPT         3600    0      86400
VDTIMEOUT     10      3      30
VDUNITJOBS    100     50     1000
VDUNITMAX     100     50     10000
VDWRITEBACK   0      0      1
* --- End of DAEXTN additions for Tunable Parameters ---
ILDMAXOPENS   OBSOLETE

```

```

ILDMAXPPA     32      16     256
* Byn Tuneable Parameters -----
BYN_MAXNETS    2      1      8
BYN_BOOTNETWAIT 30000  100    60000
BYN_MAXNETWAIT 300     100    60000
BYN_SELCOORDWAIT 1500  100    60000
BYN_FASTTIME   1      0      1
GTW_MSS_HI_WATER 0x40000 0x10000 0x80000
GTW_MSS_LO_WATER 0x2000 0x2000 0x4000
GTW_RECON_TIMEOUT 20     20     60

*****

        Unix stune file

*****

*ident "@(#)master:master.d/stune 1.4.1.4"
* Initial tuning performed on      Wed Feb  6 11:19:29 EST 2002
ARG_MAX        20480
BUFHWM 12300
DESFREE        256
HDATLIM        0x4000000
HSTKLIM        0x4000000
LOTSFREE       512
MAXUP          400
MINFREE        128
NCALL          500
NHBUF          256
NPBUF          1024
NPROC          8000
NUMTIM         3000
NUMTRW         3000
SDATLIM        0x4000000
SEMMNI         50
SEMMNS         250
SEMMNU         2500
SEMMSL         40
SSTKLIM        0x4000000
STRTHRESH      0x2800000
TRW_LOWATER    65535
UFSNINODE      4500
SHMMAX 2621440
MSGTQL         512
MSGMNB 65535
MSGMNI         512
MSGSEG 30000

```

```

VX_NINODE      10000
VDMPPGCPUCNT  2
AFFIN_ON       1
RAIO_TOGGLE    1
SEGMAPSZ       8192
SANITYCLK      0
MSGMAP         1600
MSGMAX         32767
PUTBUFSZ       20000
MSGSSZ         8
SVMMLIM        0x4000000
HVMMLIM        0x4000000
SFNOLIM        1024
SOCK_HIWATER   16384
SOCK_LOWATER   16380
BDEV_RESERVE   50
CDEV_RESERVE   50
FMOD_RESERVE   55
VFS_RESERVE    50
BYN_MAXNETS    2
BYN_BOOTNETWAIT 30000
BYN_MAXNETWAIT 300
BYN_SELCOORDWAIT 1500
BYN_FASTTIME   1
KDBSYMSIZE    200000

```

```

6. (Reserved for future use)
7. (Reserved for future use)
8. SessionMode           = 0 (Teradata)
9. LockLogger            = FALSE
10. RollbackPriority      = FALSE
11. MaxLoadTasks         = 5
12. RollForwardLock      = FALSE
13. MaxDecimal           = 18
14. Century Break        = 0
15. DateForm             = 0 (IntegerDate)
16. System TimeZone Hour = 0
17. System TimeZone Minute = 0
18. RollbackRSTransaction = FALSE
19. RSDeadLockInterval   = 0 (240)
20. RoundHalfwayMagUp    = FALSE
21. (Reserved for future use)
22. Target Level Emulation = FALSE
23. Export Width Table ID = 0 (Expected Defaults)
24. (Reserved for future use)
25. (Reserved for future use)
26. Single Sign On       = 0 (On)
27. IdCol Batch Size     = 1000 (Expected Defaults)

```

A.2 Database Parameters

Teradata DBMS Tuneables

DBS Control Record - General Fields:

```

1. Version           = 5
2. SysInit           = TRUE
3. DeadLockTimeout   = 240
4. (Reserved for future use)
5. HashFuncDBC       = 5 (Universal)

```

DBS Control Record - File System Fields:

```

1. FreeSpacePercent = 40%
2. MiniCylPackLowCylProd = 10 (free cylinders)
3. PermDBSize       = 255 (sectors)
4. JournalDBSize    = 12 (sectors)
5. DefragLowCylProd = 100 (free cylinders)
6. PermDBAllocUnit  = 1 (sectors)
7. WriteDBsToDisk   = FALSE
8. Cylinders Saved for PERM = 10 (cylinders)

```

DBS Control Record - Performance Fields:

```

1. DictionaryCacheSize = 1024 (kilobytes)
2. DBSCacheCtrl        = TRUE
3. DBSCacheThr         = 10%
4. MaxParseTreeSegs    = 1000
5. ReadAhead           = TRUE

```

```

6. StepsSegmentSize      = 1024 (kilobytes)
7. RedistBufSize         = 4 (kilobytes) *defaulted* (amp-
level buffering only)
8. DisableSyncScan       = FALSE
9. SyncScanCacheThr      = 10%
10. HTMemAlloc           = 0%
11. SkewAllowance        = 75%
12. Read Ahead Count     = 1
13. PPICacheThrP         = 10

```

```

,L_PARTKEY               INTEGER not null
,L_SUPPKEY               INTEGER not null
,L_LINENUMBER           INTEGER not null
,L_QUANTITY              DECIMAL(15,2) not null
,L_EXTENDEDPRIE         DECIMAL(15,2) not null
,L_DISCOUNT            DECIMAL(15,2) not null
,L_TAX                   DECIMAL(15,2) not null
,L_RETURNFLAG           CHAR(1) CASESPECIFIC not null
,L_LINESTATUS           CHAR(1) CASESPECIFIC not null
,L_SHIPDATE              DATE FORMAT 'yyyy-mm-dd' not null
,L_COMMITDATE           DATE FORMAT 'yyyy-mm-dd' not null
,L_RECEIPTDATE          DATE FORMAT 'yyyy-mm-dd' not null
,L_SHIPINSTRUCT         CHAR(25) CASESPECIFIC not null
,L_SHIPMODE             CHAR(10) CASESPECIFIC not null
,L_COMMENT              VARCHAR(44) CASESPECIFIC not null
PRIMARY INDEX (l_orderkey) PARTITION BY (RANGE_N
( L_ShipDate Between
* AND DATE '1992-12-31',
DATE '1993-01-01' AND DATE '1993-12-31',
DATE '1994-01-01' AND DATE '1994-12-31',
DATE '1995-01-01' AND DATE '1995-12-31',
DATE '1996-01-01' AND DATE '1996-12-31',
DATE '1997-01-01' AND DATE '1997-12-31',
DATE '1998-01-01' AND * ));

```

Appendix B. Database Definition Statements

B.1 Create Table Statements

The following CREATE User (Database) and Table statements were used to define tables used in the TPC-H benchmark test.

```

CREATE USER tpcd10000g AS PERM=40000E9,
PASSWORD=tpcd10000g;

```

```

CREATE MULTISET TABLE LINEITEM, DATABLOCKSIZE=112
KILOBYTES
(
L_ORDERKEY               DECIMAL (15,0) not null

```

```

CREATE MULTISET TABLE ORDERTBL, DATABLOCKSIZE=112 KILOBYTES
(
O_ORDERKEY              DECIMAL (15,0) not null
,O_CUSTKEY              INTEGER not null
,O_ORDERSTATUS          CHAR(1) CASESPECIFIC not null
,O_TOTALPRICE           DECIMAL(15,2) not null
,O_ORDERDATE            DATE FORMAT 'yyyy-mm-dd' not null
,O_ORDERPRIORITY        CHAR(15) CASESPECIFIC not null
,O_CLERK                CHAR(15) CASESPECIFIC not null
,O_SHIPPRIORITY         INTEGER not null
,O_COMMENT              VARCHAR(79) CASESPECIFIC not null)
PRIMARY INDEX( O_ORDERKEY ) PARTITION BY (RANGE_N
( O_OrderDate Between
* AND DATE '1992-12-31',
DATE '1993-01-01' AND DATE '1993-12-31',
DATE '1994-01-01' AND DATE '1994-12-31',
DATE '1995-01-01' AND DATE '1995-12-31',
DATE '1996-01-01' AND DATE '1996-12-31',
DATE '1997-01-01' AND DATE '1997-12-31',
DATE '1998-01-01' AND * ));

```



```

CREATE MULTISET TABLE PARTSUPP
(
  PS_PARTKEY      INTEGER not null
,PS_SUPPKEY      INTEGER not null
,PS_AVAILQTY     INTEGER not null
,PS_SUPPLYCOST   DECIMAL(15,2) not null
,PS_COMMENT      VARCHAR(199) CASESPECIFIC not null
)
;

```

```

CREATE MULTISET TABLE PARTTBL
(
  P_PARTKEY      INTEGER not null
,P_NAME         VARCHAR(55) CASESPECIFIC not null
,P_MFGR         CHAR(25) CASESPECIFIC not null
,P_BRAND        CHAR(10) CASESPECIFIC not null
,P_TYPE         VARCHAR(25) CASESPECIFIC not null
,P_SIZE         INTEGER not null
,P_CONTAINER    CHAR(10) CASESPECIFIC not null
,P_RETAILPRICE  DECIMAL(15,2) not null
,P_COMMENT      VARCHAR(23) CASESPECIFIC not null
)
UNIQUE PRIMARY INDEX( P_PARTKEY );

```

```

CREATE MULTISET TABLE SUPPLIER
(
  S_SUPPKEY      INTEGER not null
,S_NAME         CHAR(25) CASESPECIFIC not null
,S_ADDRESS      VARCHAR(40) CASESPECIFIC not null
,S_NATIONKEY    INTEGER not null
,S_PHONE        CHAR(15) CASESPECIFIC not null
,S_ACCTBAL      DECIMAL(15,2) not null
,S_COMMENT      VARCHAR(101) CASESPECIFIC not null
)
UNIQUE PRIMARY INDEX( S_SUPPKEY );

```

```

CREATE MULTISET TABLE CUSTOMER
(
  C_CUSTKEY      INTEGER not null
,C_NAME         VARCHAR(25) CASESPECIFIC not null
,C_ADDRESS      VARCHAR(40) CASESPECIFIC not null
,C_NATIONKEY    INTEGER not null
,C_PHONE        CHAR(15) CASESPECIFIC not null
,C_ACCTBAL      DECIMAL(15,2) not null

```

```

,C_MKTSEGMENT  CHAR(10) CASESPECIFIC not null
,C_COMMENT      VARCHAR(117) CASESPECIFIC not null
)
UNIQUE PRIMARY INDEX( C_CUSTKEY );

```

```

CREATE MULTISET TABLE NATION
(
  N_NATIONKEY   INTEGER NOT NULL
,N_NAME        CHAR(25) CASESPECIFIC NOT NULL
,N_REGIONKEY   INTEGER NOT NULL
,N_COMMENT     VARCHAR(152) CASESPECIFIC NOT NULL
)
UNIQUE PRIMARY INDEX( N_NAME )
;

```

```

CREATE MULTISET TABLE REGION
(
  R_REGIONKEY   INTEGER NOT NULL
,R_NAME        CHAR(25) CASESPECIFIC NOT NULL
,R_COMMENT     VARCHAR(152) CASESPECIFIC NOT NULL
)
UNIQUE PRIMARY INDEX( R_REGIONKEY );

```

B.2 Create Index and Collect Statistics Statements

```

/* This is the initial PREPARATION H index and stats */

sel date,time;

collect statistics lineitem column l_partkey;
collect statistics lineitem column l_suppkey;
collect statistics lineitem column l_orderkey;
collect statistics lineitem column l_shipdate;
collect statistics lineitem column l_linenumber;

sel date,time;

create unique index ox_ok (o_orderkey) on ordertbl;
collect statistics ordertbl column o_orderkey;
collect statistics ordertbl column o_custkey;
collect statistics ordertbl column o_orderdate;
collect statistics ordertbl column o_shippriority;

```

```

sel date,time;

create index psx_sk (ps_suppkey) on partsupp;
collect statistics partsupp column ps_partkey;
collect statistics partsupp column ps_suppkey;
collect statistics partsupp column ps_availqty;

sel date,time;

collect statistics parttbl column p_partkey;
collect statistics parttbl column p_type;
collect statistics parttbl column p_name;
collect statistics parttbl column p_size;
collect statistics parttbl column p_comment;

sel date,time;

create index sx_nk (s_nationkey) on supplier;
collect statistics supplier column s_suppkey;
collect statistics supplier column s_nationkey;
collect statistics supplier column s_name;

sel date,time;

create index cx_nk (c_nationkey) on customer;
collect statistics customer column c_custkey;
collect statistics customer column c_nationkey;
collect statistics customer column c_name;

sel date,time;

collect statistics region column r_regionkey;
collect statistics region column r_name;

sel date,time;

collect statistics nation column n_nationkey;
collect statistics nation column n_name;
collect statistics nation column n_regionkey;

sel date,time;

alter table lineitem, default datablocksize;
alter table ordertbl, default datablocksize;

sel date,time;

```

```

.quit

.set width 132
sel date,time;
.os /home2/tpcd/preps/modify_DBs

create hash index l_fpk2 (l_partkey,l_suppkey) on lineitem by
(l_partkey) order by hash (l_partkey);

.os /home2/tpcd/preps/modify_defaultDBs
sel date,time;

.quit

.set width 132
sel date,time;
.os /home2/tpcd/preps/modify_DBs

create hash index o_fpk2 (o_custkey) on ordertbl by (o_custkey)
order by hash (o_custkey);

.os /home2/tpcd/preps/modify_defaultDBs
sel date,time;

.quit

```

Appendix C. Query Text and Output

Data from Qualification Database

Submitting SQL Request #1:

```

/* Teradata
*/
/* @(#)TERADATA 1.sql 1.1.1.1@(#) */
/* Query 01 - Var_0 Rev_01 - TPC-H/TPC-R Pricing Summary Report
Query */

SELECT
    L_RETURNFLAG,
    L_LINESTATUS,
    SUM(L_QUANTITY (FLOAT)) (DECIMAL(18,2)) AS SUM_QTY,
    SUM(L_EXTENDEDPRI (FLOAT)) (DECIMAL(18,2)) AS
SUM_BASE_PRICE,
    SUM(L_EXTENDEDPRI*(1-L_DISCOUNT) (FLOAT))
(DECIMAL(18,2)) AS SUM_DISC_
PRICE,
    SUM(L_EXTENDEDPRI*(1-L_DISCOUNT)*(1+L_TAX) (FLOAT))
(DECIMAL(18,2)) AS
SUM_CHARGE,
    AVG(L_QUANTITY (FLOAT)) (DECIMAL(18,2)) AS AVG_QTY,
    AVG(L_EXTENDEDPRI (FLOAT)) (DECIMAL(18,2)) AS
AVG_PRICE,
    AVG(L_DISCOUNT (FLOAT)) (DECIMAL(18,2)) AS AVG_DISC,
    COUNT(*) (DEC(18,0)) AS COUNT_ORDER
FROM
    LINEITEM
WHERE
    L_SHIPDATE <= DATE '1998-12-01' - INTERVAL '90' DAY
GROUP BY
    L_RETURNFLAG,
    L_LINESTATUS
ORDER BY
    L_RETURNFLAG,

```

L_LINESTATUS;

Query 1 complete, 4 rows returned

Col1	Col2	Col3	Col4	Col5	Col6	Col7	Col8	Col9	Col10
A	F	37734107.00	56586554400.73	53758257134.87	55909065222.83	25.52	38273.13	0.05	1478493.00
N	F	991417.00	1487504710.38	1413082168.05	1469649223.19	25.52	38284.47	0.05	38854.00
N	O	74476040.00	111701729697.74	106118230307.61	110367043872.50	25.50	38249.12	0.05	2920374.00
R	F	37719753.00	56568041380.90	53741292684.60	55889619119.83	25.51	38250.85	0.05	1478870.00

Submitting SQL Request #2:

```

/* Teradata
*/
/* @(#)TERADATA 2.sql 1.1.1.1@(#) */
/* Query 02 - Var_0 Rev_01 - TPC-H/TPC-R Minimum Cost Supplier
Query */

SELECT
    S_ACCTBAL,
    S_NAME,
    N_NAME,
    P_PARTKEY,
    P_MFGR,
    S_ADDRESS,
    S_PHONE,
    S_COMMENT

```

```

FROM
PARTTBL,
SUPPLIER,
PARTSUPP,
NATION,
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
PARTSUPP,
SUPPLIER,
NATION,
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;

```

Query 2 complete, 460 rows returned

only displaying 100 rows per TPC-H spec

```

|_____Col1|Col2_____|Col3_____
|_____
Col4|Col5_____|Col6_____
|_____|Col7_____
|_____|Col8_____
|_____
|_____

```

```

|          9938.53|Supplier#000005359      |UNITED KINGDOM
|          18
5358|Manufacturer#4          |QKuHYh,vZGiwu2FWEJoLDx04
|33-429-7
90-6131|blithely silent pinto beans are furiously. slyly final
deposits across
|          9937.84|Supplier#000005969      |ROMANIA
|          10
8438|Manufacturer#1          |ANDENSOSmk,miq23Xfb5RWt6dvUcvt6Qa
|29-520-6
92-3537|carefully slow deposits use furiously. slyly ironic
platelets above the
ironic |
|          9936.22|Supplier#000005250      |UNITED KINGDOM
|          249|Manufacturer#4          |B3rqp0xbSEim4Mpy2RH J
|33-320-2
28-2957|blithely special packages are. stealthily express
deposits across the cl
osely final instructi|
|          9923.77|Supplier#000002324      |GERMANY
|          2
9821|Manufacturer#4          |y3OD9UywSTok
|17-779-2
99-1839|quickly express packages breach quiet pinto beans. requ
|
|          9871.22|Supplier#000006373      |GERMANY
|          4
3868|Manufacturer#5          |J8fcXWsTqM
|17-813-4
85-8637|never silent deposits integrate furiously blit
|
|          9870.78|Supplier#000001286      |GERMANY
|          8
1285|Manufacturer#2
|YKA,E2fjiVd7eUrzp2Ef8j1QxGo2DFnosATEH |17-516-9
24-4574|final theodolites cajole slyly special,
|
|          9870.78|Supplier#000001286      |GERMANY
|          18
1285|Manufacturer#4
|YKA,E2fjiVd7eUrzp2Ef8j1QxGo2DFnosATEH |17-516-9
24-4574|final theodolites cajole slyly special,
|
|          9852.52|Supplier#000008973      |RUSSIA
|          1

```

8972|Manufacturer#2 |t5L67YdBYYH6o,Vz24jpDyQ9
|32-188-5
94-7038|quickly regular instructions wake-- carefully unusual
braids into the ex
pres |
| 9847.83|Supplier#000008097 |RUSSIA
| 13
0557|Manufacturer#2 |xMe97bpE69NzdWLoX
|32-375-6
40-3593|slyly regular dependencies sleep slyly furiously express
dep
|
| 9847.57|Supplier#000006345 |FRANCE
| 8
6344|Manufacturer#1
|VSt3rzK3qG698u6ld8HhOByvrTcSTsvQldQDag |16-886-7
66-7945|silent pinto beans should have to snooze carefully along
the final reque
s |
| 9847.57|Supplier#000006345 |FRANCE
| 17
3827|Manufacturer#2
|VSt3rzK3qG698u6ld8HhOByvrTcSTsvQldQDag |16-886-7
66-7945|silent pinto beans should have to snooze carefully along
the final reque
s |
| 9836.93|Supplier#000007342 |RUSSIA
|
4841|Manufacturer#4 |J0lK7C1,7xrEZSSow
|32-399-4
14-5385|final accounts haggle. bold accounts are furiously
dugouts. furiously si
lent asymptotes are slyly|
| 9817.10|Supplier#000002352 |RUSSIA
| 12
4815|Manufacturer#2 |4LfoHUZjgjEbAKw TgdKcgOc4D4uCYw
|32-551-8
31-1437|blithely pending packages across the ironic accounts grow
slyly after th
e furiously|
| 9817.10|Supplier#000002352 |RUSSIA
| 15
2351|Manufacturer#3 |4LfoHUZjgjEbAKw TgdKcgOc4D4uCYw
|32-551-8
31-1437|blithely pending packages across the ironic accounts grow
slyly after th
e furiously|

| 9739.86|Supplier#000003384 |FRANCE
| 13
8357|Manufacturer#2 |o,Z3v4POifevE k9U1b 6JlucX,I
|16-494-9
13-5925|slyly ironic theodolites hag
|
| 9721.95|Supplier#000008757 |UNITED KINGDOM
| 15
6241|Manufacturer#3 |Atg6GnM4dT2
|33-821-4
07-2995|ironic, even dolphins above the furiously ironic foxes
sleep slyly aroun
d the caref|
| 9681.33|Supplier#000008406 |RUSSIA
| 7
8405|Manufacturer#1 |,qUuXcftU1
|32-139-8
73-8571|furiously even deposits affix thinly special theodolites.
furiou
|
| 9643.55|Supplier#000005148 |ROMANIA
| 10
7617|Manufacturer#1 |kT4ciVFslx9z4s79p Js825
|29-252-6
17-4850|doggedly even ideas boost furiously against the furiously
express
|
| 9624.82|Supplier#000001816 |FRANCE
| 3
4306|Manufacturer#3 |e7vab91vLJPWxxZnewmDBpDmxYHrb
|16-392-2
37-6726|blithely regular accounts cajole furiously. regular
|
| 9624.78|Supplier#000009658 |ROMANIA
| 18
9657|Manufacturer#1 |oE9uBgEfSS4opIcepXyAYM,x
|29-748-8
76-2014|regular deposits haggle. furiously express asympto
|
| 9612.94|Supplier#000003228 |ROMANIA
| 12
0715|Manufacturer#2 |KDDpNKN3cWu7ZSrbdq7AfSLxx,qWB
|29-325-7
84-8187|carefully pending accounts serve. furiously close
deposits boost slyly.
q |

	9612.94	Supplier#000003228	ROMANIA	2349	Manufacturer#5	HYogcF3Jb yh1
	19				29-334-8	
8189	Manufacturer#4	KDDpNKN3cWu7zSrbdqp7AfSLxx,qWB		70-9731	carefully unusual packages sleep carefully even ideas.	
	29-325-7				dogged accoun	
84-8187	carefully pending accounts serve. furiously close					
deposits boost slyly.					9357.45	Supplier#000006188 UNITED KINGDOM
q					13	
	9571.83	Supplier#000004305	ROMANIA	8648	Manufacturer#1	g801,ssP8wpTk4Hm
	17				33-583-6	
9270	Manufacturer#2	qNHZ7WmCzygwMPRDO9Ps		07-1633	carefully regular deposits wake carefully furiously even	
	29-973-4			i		
81-1831	furiously final deposits				9352.04	Supplier#000003439 GERMANY
					17	
	9558.10	Supplier#000003532	UNITED KINGDOM	0921	Manufacturer#4	qYPDgoiBGhCYxjgC
	8				17-128-9	
8515	Manufacturer#4	EOeuiiOn2lOVpTlGguufFDFsbN1p0lhpxHp	33-152-3	96-4650	fluffily regular pinto beans wake. unusual, final ideas c	
	01-2164	daring, sly accounts breach about th				
					9312.97	Supplier#000007807 RUSSIA
	9492.79	Supplier#000005975	GERMANY		9	
	2			0279	Manufacturer#5	oGYMPck9XHGB2PBfKRnHA
5974	Manufacturer#5	S6mIiCTx82z7lV			32-673-8	
	17-992-5			72-5854	unusual asymptotes above the	
79-4839	always pending packages boost slyly.					
					9312.97	Supplier#000007807 RUSSIA
	9461.05	Supplier#000002536	UNITED KINGDOM		10	
	2			0276	Manufacturer#5	oGYMPck9XHGB2PBfKRnHA
0033	Manufacturer#1	8mmGbyzaU			32-673-8	
7ZS2wJumTibypncu9pNkDc4FYA	33-556-9			72-5854	unusual asymptotes above the	
73-5522	even foxes are quickly furiously express requests.					
packages					9280.27	Supplier#000007194 ROMANIA
					4	
	9453.01	Supplier#000000802	ROMANIA	7193	Manufacturer#3	zhRUQkBSrFYxIAXTfInj vyGRQjeK
	17				29-318-4	
5767	Manufacturer#1	,6HYXb4uaHITmtMBj4Ak57Pd		54-2133	slyly ironic requests despite the unusual ins	
	29-342-8					
82-6463	final, regular packages across the slowly regular packag				9274.80	Supplier#000008854 RUSSIA
					7	
	9408.65	Supplier#000007772	UNITED KINGDOM	6346	Manufacturer#3	1xhLoOUM7I3mZ1mKnerw OSqdbb4QbGa
	11				32-524-1	
7771	Manufacturer#4	AiC5YAH,gdu0i7		48-5221	ruthlessly ironic instructions along the regular, furious	
	33-152-4			requests integ		
91-1126	blithely final ideas sleep carefully. requests are			rate car		
					9249.35	Supplier#000003973 FRANCE
	9359.61	Supplier#000004856	ROMANIA		2	
	6			6466	Manufacturer#1	dl8GiDsL6Wm2IsGXM,RZf1jCsgZAOjNYVThTRP4 16-722-8

66-1658 quickly ironic sauternes use b		9101.00 Supplier#000005791	ROMANIA
		12	
	9249.35 Supplier#000003973	FRANCE	
	3		
3972 Manufacturer#1		8254 Manufacturer#5	zub2zCV,jhHPPQqi,P2INAJE1zI
dl8GiDsL6Wm2IsGXM,RZfljCsgZAOjNYVThTRP4	16-722-8	n66cOEoXFG	29-549-2
66-1658 quickly ironic sauternes use b		51-5384 carefully ironic packages after the	
	9208.70 Supplier#000007769	ROMANIA	
	4		
0256 Manufacturer#5	rsimdze 5o9P Ht7xS	9575 Manufacturer#1	WB0XkCSG3r,mnQ
29-964-4		n,h9Vixj9ARHFvKgMDF	32-587-5
24-9649 furiously ruthless epitaphs among the furiously regular		77-1351 asymptotes above the slyly even requests haggle furiously	
accounts use slo		about the regu	
wly fluffily ev		lar accounts	
	9201.47 Supplier#000009690	UNITED KINGDOM	
	6		
7183 Manufacturer#5	CB BnUTlmi5zdeE17R7	2191 Manufacturer#5	8XVcQK23akp
33-121-2		16-811-2	
67-9529 blithely unusual accounts integrate slyly. platelets		69-8946 stealthy requests haggle c	
	9192.10 Supplier#000000115	UNITED KINGDOM	
	8		
5098 Manufacturer#3	nJ 2t0f7Ve,wL1,6WzGBJLNBUCKlsV	9813 Manufacturer#2	af005pg831PU4IDVmEylXZVqYZQzSDlYLA mR
33-597-2		29-995-5	
48-1220 slyly bold pinto beans boost across the furiously regular		71-8781 ironic theodolites are evenly unusual requests-- pending	
packages. care		pinto beans acr	
fully regu		oss the in	
	9189.98 Supplier#000001226	GERMANY	
	2		
1225 Manufacturer#4	qsLCqSvLyZfuXIpjz		
17-725-9		8968.42 Supplier#000010000	ROMANIA
03-1381 final, express instruction			
		11	
	9128.97 Supplier#000004311	RUSSIA	
	14		
6768 Manufacturer#5	I8IjnXd7NSJRs594RxsRR0	9999 Manufacturer#5	aTGLEusCiL4F PDBdv665XBjHPyCOB0i
32-155-4		29-578-4	
40-7120 regular pinto beans sleep ca		32-2146 furiously final ideas believe furiously. furiously final	
		ideas	
	9104.83 Supplier#000008520	UNITED KINGDOM	
	15		
0974 Manufacturer#4	RqRVDgD0ER J9 b41vR2,3	9512 Manufacturer#1	FVajceZInZdbJE6Z9XsRUxrUEpiwHDrOXi,1Rz
17-728-8		33-784-1	
04-1793 deposits sleep carefully e		77-8208 furiously regular excuses wake after the blithely special	
		pinto beans? e	
		ven instructions sl	
		8929.42 Supplier#000008770	FRANCE
		17	
0974 Manufacturer#4	R7cG26TtXrHAP9 Hckhfri	3735 Manufacturer#4	
17-728-8		16-242-7	
04-1793 deposits sleep carefully e		46-9248 final accounts sleep furiously. blithely ironic foxes	
		wake boldly across	
		the furiously s	

	8920.59	Supplier#000003967	ROMANIA	9406	Manufacturer#4	CHRCbkaWcf5B
	2				33-903-9	
6460	Manufacturer#1	eHoAXe62SY9		70-9604	regular dependencies haggle across the carefully bold	
	29-194-7					
31-3944	quickly even requests should have to affix blithely-- fur				8691.06	Supplier#000004429 UNITED KINGDOM
					12	
	8920.59	Supplier#000003967	ROMANIA	6892	Manufacturer#2	k,BQms5UhoAF1B2Asi,fLib
	17				33-964-3	
3966	Manufacturer#2	eHoAXe62SY9		37-5038	quickly special foxes against the furiously silent	
	29-194-7				platelets wake quickl	
31-3944	quickly even requests should have to affix blithely-- fur			y after t		
					8655.99	Supplier#000006330 RUSSIA
	8913.96	Supplier#000004603	UNITED KINGDOM		19	
	13			3810	Manufacturer#2	UozlaENr0ytKe2w6CeIEWFWn
7063	Manufacturer#2	OUzlvMUr7n,utLxmpNeYKsf3T24OXskxB5		iO3S8Rae7Ou	32-561-1	
	33-789-2			98-3705	blithely even packages alongside	
55-7342	slyly ironic packages detect furious accounts. ironic de					
					8638.36	Supplier#000002920 RUSSIA
	8877.82	Supplier#000007967	FRANCE		7	
	16			5398	Manufacturer#1	Je2a8bszf3L
7966	Manufacturer#5	A3pilBARM4nx6R,qrwFoRPU			32-122-6	
	16-442-1			21-7549	express deposits wake. furiously silent requests wake	
47-9345	final deposits after the silent deposits ha				carefully silent i	
				nstru		
	8862.24	Supplier#000003323	ROMANIA		8638.36	Supplier#000002920 RUSSIA
	7				17	
3322	Manufacturer#3	W9 lYcsC9FwBqk3ItL		0402	Manufacturer#3	Je2a8bszf3L
	29-736-9				32-122-6	
51-3710	unusual, pending theodolites integrate furiously slyly			21-7549	express deposits wake. furiously silent requests wake	
even pinto beans.					carefully silent i	
unusual sheaves sleep befor				nstru		
	8841.59	Supplier#000005750	ROMANIA		8607.69	Supplier#000006003 UNITED KINGDOM
	10				7	
0729	Manufacturer#5	Erx3lAgu0g62iaHF9x50uMH4EgeN9hEG		6002	Manufacturer#2	
	29-344-5				EH9wADcEiuenM0NR08zDwMidw,52Y2RyILEiA	33-416-8
02-5481	excuses after the blithely regular packages mold			07-5206	always special foxes wake slyly bold, ironic accounts.	
carefully deposits. reg					ironic instructio	
ular a				ns affix carefull		
	8781.71	Supplier#000003121	ROMANIA		8569.52	Supplier#000005936 RUSSIA
	1					
3120	Manufacturer#5	wNqTogx238ZYCamFb,50v,bj		5935	Manufacturer#5	jXaNZ6vwnEWJ2ksLZJpjtgt0bY2a3AU
	4IbNFW9Bvw1xP	29-707-2			32-644-2	
91-5144	packages are quickly after the final, even packages.			51-7916	packages sleep furiously. special requests about the	
furiously regular					fluffily even accou	
				nts detect		
	8754.24	Supplier#000009407	UNITED KINGDOM		8564.12	Supplier#000000033 GERMANY
	17				11	

0032|Manufacturer#1
|gfeKpYw3400L0SDywXA6YalQmqlw6YB9f3R |17-138-8
97-9374|ironic instructions are. special pearls above
|
| 8553.82|Supplier#000003979 |ROMANIA
| 14
3978|Manufacturer#4 |BfmVhCAnCMY3jzpjUMy4CNWs9
HzpdQR7INJU |29-124-6
46-4897|express, ironic pinto beans cajole around the express,
even packages. qu
|
| 8517.23|Supplier#000009529 |RUSSIA
| 3
7025|Manufacturer#5 |e44R8o7JAIS9iMcr
|32-565-2
97-8775|furiously silent requests cajole furiously furiously
ironic foxes. slyly
express p|
| 8517.23|Supplier#000009529 |RUSSIA
| 5
9528|Manufacturer#2 |e44R8o7JAIS9iMcr
|32-565-2
97-8775|furiously silent requests cajole furiously furiously
ironic foxes. slyly
express p|
| 8503.70|Supplier#000006830 |RUSSIA
| 4
4325|Manufacturer#4 |BC4WFCYRUZyaIgchU 4S
|32-147-8
78-5069|quickly regular excuses detect evenly around
|
| 8457.09|Supplier#000009456 |UNITED KINGDOM
| 1
9455|Manufacturer#1 |7SBhZs8gPlcJjT0Qf433YBk
|33-858-4
40-4349|carefully final accounts sleep blithely special foxes.
slyly regular pin
to beans alon|
| 8441.40|Supplier#000003817 |FRANCE
| 14
1302|Manufacturer#2 |hU3fz3xL78
|16-339-3
56-5115|blithely blithe ideas are
|
| 8432.89|Supplier#000003990 |RUSSIA
| 19

1470|Manufacturer#1
|wehBBp1RQbfxAyDASS75MsywmsKHRVdkrvNe6m |32-839-5
09-9301|final requests along the blithely ironic packages kindle
against the car
efully fina|
| 8431.40|Supplier#000002675 |ROMANIA
|
5174|Manufacturer#1 |HJFStOu9R5NGPOegKhgbzBdyvrG2yh8w
|29-474-6
43-1443|express, final deposits cajole carefully. stealthily
unusual requests
|
| 8407.04|Supplier#000005406 |RUSSIA
| 16
2889|Manufacturer#4 |j7 gYF5RW8DC5UrjKC
|32-626-1
52-4621|quickly final sheaves boost. car
|
| 8386.08|Supplier#000008518 |FRANCE
| 3
6014|Manufacturer#3
|2jqzqqAVe9crMVGP,n9nTsQXulNLTUYoJjEDcqWV|16-618-7
80-7481|slyly ironic theodolites are slyly. dogged, pendin
|
| 8376.52|Supplier#000005306 |UNITED KINGDOM
| 19
0267|Manufacturer#5 |9t8Y8
QqSIsoADPt6NLdk,TP5zyRx41oBUlgoGc9|33-632-5
14-7931|furiously even instructions integrate during the
furiously regular re
|
| 8348.74|Supplier#000008851 |FRANCE
| 6
6344|Manufacturer#4 |nWxi7GwEbjhw1
|16-796-2
40-2472|ironic instructions nag slyly against the slyly even
theodolites. reques
ts alongside of |
| 8338.58|Supplier#000007269 |FRANCE
| 1
7268|Manufacturer#4 |ZwhJSwABUoiB04,3
|16-267-2
77-4365|ruthlessly regular asymptotes a
|
| 8328.46|Supplier#000001744 |ROMANIA
| 6

9237|Manufacturer#5 |oLo3fV64q2,FKHa3p,qHnS7Yzv,ps8
|29-330-7
28-5873|blithely silent excuses are slyly above the furiously
even courts
|
| 8307.93|Supplier#000003142 |GERMANY
| 1
8139|Manufacturer#1 |dqblvV8dCNAorGlJ
|17-595-4
47-6026|theodolites sleep blithely carefully regular warhorses.
slyly regular in
s |
| 8231.61|Supplier#000009558 |RUSSIA
| 19
2000|Manufacturer#2 |mcdgen,yTliJDHDS5fv
|32-762-1
37-5858|slyly regular theodolites sleep fluffily express depos
|
| 8152.61|Supplier#000002731 |ROMANIA
| 1
5227|Manufacturer#4 |nluXJCuYltu
|29-805-4
63-2030|gifts use. slyly silent ideas are carefully beneath the
silent instructi
ons. slyly sil|
| 8109.09|Supplier#000009186 |FRANCE
| 9
9185|Manufacturer#1 |wgfosrVPexl9pEXWywaqlBMDYYf
|16-668-5
70-1402|quickly pending requests are blithely along the ironic,
final requests;
instr |
| 8102.62|Supplier#000003347 |UNITED KINGDOM
| 1
8344|Manufacturer#5 |m CtXS2S16i
|33-454-2
74-8532|packages grow special orbits. regular theodolites about
the carefully pe
|
| 8046.07|Supplier#000008780 |FRANCE
| 19
1222|Manufacturer#3 |AczzuE0UK9osj ,Lx0Jmh
|16-473-2
15-6395|regular epitaphs integrate slyly.
|
| 8042.09|Supplier#000003245 |RUSSIA
| 13

5705|Manufacturer#4 |Dh8Ikg39onrbOL4DyTfGw8a9oKUX3d9Y
|32-836-1
32-8872|carefully regular instructions integrate blithely silent
foxes. furiousl
y express instructions hagg|
| 8042.09|Supplier#000003245 |RUSSIA
| 15
0729|Manufacturer#1 |Dh8Ikg39onrbOL4DyTfGw8a9oKUX3d9Y
|32-836-1
32-8872|carefully regular instructions integrate blithely silent
foxes. furiousl
y express instructions hagg|
| 7992.40|Supplier#000006108 |FRANCE
| 11
8574|Manufacturer#1 |8tBydnTDWUqfBfFV413
|16-974-9
98-8937|regular pinto beans are after
|
| 7980.65|Supplier#000001288 |FRANCE
| 1
3784|Manufacturer#4 |zE,7HgVPrCn
|16-646-4
64-8247|unusual pinto beans cajole furiously according t
|
| 7950.37|Supplier#000008101 |GERMANY
| 3
3094|Manufacturer#5 |kkYvL6IuvojJgTNG IKkaXQDYgx8ILohj
|17-627-6
63-8014|quickly regular requests are furiously. pending deposits
wake
|
| 7937.93|Supplier#000009012 |ROMANIA
| 8
3995|Manufacturer#2 |iUiTziH,Ek3i4lwSgunXMgrcTzwdb
|29-250-9
25-9690|blithely bold ideas haggle quickly final, regular request
|
| 7914.45|Supplier#000001013 |RUSSIA
| 12
5988|Manufacturer#2 |riRcntps4KEDtYScjpMIWeYF6mNnR
|32-194-6
98-3365|final, ironic theodolites alongside of the ironic
|
| 7912.91|Supplier#000004211 |GERMANY
| 15
9180|Manufacturer#5 |2wQRVovHrm3,v03IKzftD,1PYsFXQFFOG
|17-266-9

```

47-7315|final requests integrate slyly above the silent, even
|
|          7912.91|Supplier#000004211          |GERMANY
|          18
4210|Manufacturer#4          |2wQRVovHrm3,v03IKzfTd,1PYsFXQFFOG
|17-266-9
47-7315|final requests integrate slyly above the silent, even
|
|          7894.56|Supplier#000007981          |GERMANY
|          8
5472|Manufacturer#4          |NSJ96vMROAbeXP
|17-963-4
04-3760|regular, even theodolites integrate carefully. bold,
special theodolites
are slyly fluffily iron|
|          7887.08|Supplier#000009792          |GERMANY
|          16
4759|Manufacturer#3          |Y28ITVeYrit3kIGdV2K8fSZ
V2UqT5H10tz          |17-988-9
38-4296|pending, ironic packages sleep among the carefully ironic
accounts. quic
kly final accounts |
|          7871.50|Supplier#000007206          |RUSSIA
|          10
4695|Manufacturer#1          |3w fNCnrVmvJjE95sgWZzvW
|32-432-4
52-7731|furiously dogged pinto beans cajole. bold, express
notornis until the sl
yly pending |
|          7852.45|Supplier#000005864          |RUSSIA
|
8363|Manufacturer#4          |WCNfBPZeSXh3h,c
|32-454-8
83-3821|blithely regular deposits
|
|          7850.66|Supplier#000001518          |UNITED KINGDOM
|          8
6501|Manufacturer#1          |ONda3YJiHKJOC
|33-730-3
83-3892|furiously final accounts wake carefully idle requests.
even dolphins wak
e acc |
|          7843.52|Supplier#000006683          |FRANCE
|          1
1680|Manufacturer#4          |2Z0JGkiv01Y00oCFwUGfviIbhzcDy
|16-464-5
17-8943|carefully bold accounts doub

```

Submitting SQL Request #3:

```

/* Teradata
*/
/* @(#)TERADATA 3.sql 1.1.1.1@(#) */
/* Query 03 - Var_0 Rev_01 - TPC-H/TPC-R Shipping Priority Query
*/

SELECT
    L_ORDERKEY,
    SUM(L_EXTENDEDPRI*(1-L_DISCOUNT)) (FLOAT)
(DECIMAL(18,2)) AS REVENUE,
    O_ORDERDATE,
    O_SHIPPRIORITY
FROM
    CUSTOMER,
    ORDERTBL,
    LINEITEM
WHERE
    C_MKTSEGMENT = 'BUILDING'
    AND C_CUSTKEY = O_CUSTKEY
    AND L_ORDERKEY = O_ORDERKEY
    AND O_ORDERDATE < '1995-03-15'
    AND L_SHIPDATE > '1995-03-15'
GROUP BY
    L_ORDERKEY,
    O_ORDERDATE,
    O_SHIPPRIORITY
ORDER BY
    REVENUE DESC,
    O_ORDERDATE;

```

Query 3 complete, 11620 rows returned

only displaying 10 rows per TPC-H spec

Col1	Col2	Col3	Col4
2456423.00	406181.01	1995-03-05	
3459808.00	405838.70	1995-03-04	
492164.00	390324.06	1995-02-19	

0	1188320.00	384537.94	1995-03-09
0	2435712.00	378673.06	1995-02-26
0	4878020.00	378376.80	1995-03-12
0	5521732.00	375153.92	1995-03-13
0	2628192.00	373133.31	1995-02-22
0	993600.00	371407.46	1995-03-05
0	2300070.00	367371.15	1995-03-13

Submitting SQL Request #4:

```

/* Teradata
*/
/* @(#)TERADATA 4.sql 1.1.1.1@(#) */
/* Query 04 - Var_0 Rev_01 - TPC-H/TPC-R Order Priority Checking
Query */

```

```

SELECT
    O_ORDERPRIORITY,
    COUNT(*) AS ORDER_COUNT
FROM
    ORDERTBL
WHERE
    O_ORDERDATE >= '1993-07-01'
    AND O_ORDERDATE < DATE '1993-07-01' + INTERVAL '3' MONTH
    AND EXISTS (
        SELECT
            *
        FROM
            LINEITEM
        WHERE
            L_ORDERKEY = O_ORDERKEY
            AND L_COMMITDATE < L_RECEIPTDATE
    )
GROUP BY
    O_ORDERPRIORITY
ORDER BY
    O_ORDERPRIORITY;

```

Query 4 complete, 5 rows returned

Coll_____	_____Col2
1-URGENT	10594
2-HIGH	10476
3-MEDIUM	10410
4-NOT SPECIFIED	10556
5-LOW	10487

Submitting SQL Request #5:

```

/* Teradata
*/
/* @(#)TERADATA 5.sql 1.1.1.2@(#) */
/* Query 05 - Var_0 Rev_01 - TPC-H/TPC-R Local Supplier Volume
Query */

```

```

SELECT
    N_NAME,
    SUM(L_EXTENDEDPRIICE*(1-L_DISCOUNT) (FLOAT))
(DECIMAL(18,2)) AS REVENUE
FROM
    CUSTOMER,
    ORDERTBL,
    LINEITEM,
    SUPPLIER,
    NATION,
    REGION
WHERE
    C_CUSTKEY = O_CUSTKEY
    AND L_ORDERKEY = O_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 >= '1994-01-01'
    AND O_ORDERDATE < DATE '1994-01-01' + INTERVAL '1' YEAR
GROUP BY
    N_NAME
ORDER BY
    REVENUE DESC;

```

Query 5 complete, 5 rows returned

Coll_____	_____Col2
INDONESIA	55502041.17
VIETNAM	55295087.00
CHINA	53724494.26

```
|INDIA                |          52035512.00|
|JAPAN                |          45410175.70|
```

Submitting SQL Request #6:

```
/* Teradata
*/
/* @(#)TERADATA 6.sql 1.1.1.1@(#) */
/* Query 06 - Var_0 Rev_01 - TPC-H/TPC-R Forecasting Revenue
Change Query */

SELECT
    SUM(L_EXTENDEDPRI*L_DISCOUNT (FLOAT)) (DECIMAL(18,2))
AS REVENUE
FROM
    LINEITEM
WHERE
    L_SHIPDATE >= '1994-01-01'
    AND L_SHIPDATE < DATE '1994-01-01' + INTERVAL '1' YEAR
    AND L_DISCOUNT BETWEEN .06 - 0.01 AND .06 + 0.01
    AND L_QUANTITY < 24;
```

Query 6 complete, 1 rows returned

```
|_____Col1|
|          123141078.23|
```

Submitting SQL Request #7:

```
/* Teradata
*/
/* @(#)TERADATA 7.sql 1.1.1.1@(#) */
/* Query 07 - Var_0 Rev_01 - TPC-H/TPC-R Volume Shipping Query
*/

SELECT
    N1.N_NAME AS SUPP_NATION,
    N2.N_NAME AS CUST_NATION,
    EXTRACT(YEAR FROM L_SHIPDATE) AS "YEAR",
    SUM(L_EXTENDEDPRI * (1-L_DISCOUNT) (FLOAT))
(DECIMAL(18,2)) AS REVENUE
FROM
    SUPPLIER,
    LINEITEM,
    ORDERTBL,
    CUSTOMER,
```

```
NATION N1,
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 '1995-01-01' AND '1996-12-31'
GROUP BY
    SUPP_NATION,
    CUST_NATION,
    "YEAR"
ORDER BY
    SUPP_NATION,
    CUST_NATION,
    "YEAR";
```

Query 7 complete, 4 rows returned

```
|Col1_____|Col2_____|_____Col3|
|_____
___Col4|
|FRANCE                |GERMANY                |          1995|
5463
9732.73|
|FRANCE                |GERMANY                |          1996|
5463
3083.31|
|GERMANY                |FRANCE                |          1995|
5253
1746.67|
|GERMANY                |FRANCE                |          1996|
5252
0549.02|
```

Submitting SQL Request #8:

```
/* Teradata
*/
/* @(#)TERADATA 8.sql 1.1.1.1@(#) */
```

```
/* Query 08 - Var_0 Rev_01 - TPC-H/TPC-R National Market Share
Query */
```

```
SELECT
    EXTRACT(YEAR FROM O_ORDERDATE) AS "YEAR",
    SUM(CASE
        WHEN N2.N_NAME = 'BRAZIL'
        THEN (L_EXTENDEDPRI*(1-L_DISCOUNT) (FLOAT))
        ELSE 0
    END) / SUM(L_EXTENDEDPRI*(1-L_DISCOUNT) (FLOAT))
(DECIMAL(18,2)) AS MK
T_SHARE
FROM
    PARTTBL,
    SUPPLIER,
    LINEITEM,
    ORDERTBL,
    CUSTOMER,
    NATION N1,
    NATION N2,
    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 '1995-01-01' AND '1996-12-31'
    AND P_TYPE = 'ECONOMY ANODIZED STEEL'
GROUP BY
    "YEAR"
ORDER BY
    "YEAR";
```

Query 8 complete, 2 rows returned

Col1	Col2
1995	0.03
1996	0.04

Submitting SQL Request #9:

```
/* Teradata
*/
/* @(#)TERADATA 9.sql 1.1.1.1@(#) */
/* Query 09 - Var_0 Rev_01 - TPC-H/TPC-R Product Type Profit
Measure Query */
```

```
SELECT
    N_NAME AS NATION,
    EXTRACT(YEAR FROM O_ORDERDATE) AS "YEAR",
    SUM(L_EXTENDEDPRI*(1-L_DISCOUNT)-
    PS_SUPPLYCOST*L_QUANTITY (FLOAT)) (DE
    CIMAL(18,2)) AS SUM_PROFIT
FROM
    PARTTBL,
    SUPPLIER,
    LINEITEM,
    PARTSUPP,
    ORDERTBL,
    NATION
WHERE
    S_SUPPKEY = L_SUPPKEY
    AND PS_SUPPKEY = L_SUPPKEY
    AND PS_PARTKEY = L_PARTKEY
    AND P_PARTKEY = L_PARTKEY
    AND O_ORDERKEY = L_ORDERKEY
    AND S_NATIONKEY = N_NATIONKEY
    AND P_NAME LIKE '%green%'
GROUP BY
    NATION,
    "YEAR"
ORDER BY
    NATION,
    "YEAR" DESC;
```

Query 9 complete, 175 rows returned

Col1	Col2	Col3
ALGERIA	1998	31342867.23
ALGERIA	1997	57138193.02
ALGERIA	1996	56140140.13
ALGERIA	1995	53051469.65
ALGERIA	1994	53867582.13
ALGERIA	1993	54942718.13
ALGERIA	1992	54628034.71
ARGENTINA	1998	30211185.71
ARGENTINA	1997	50805741.75
ARGENTINA	1996	51923746.58

ARGENTINA	1995	49298625.77	GERMANY	1997	49309074.88
ARGENTINA	1994	50835610.11	GERMANY	1996	49918683.17
ARGENTINA	1993	51646079.18	GERMANY	1995	52650718.72
ARGENTINA	1992	50410314.99	GERMANY	1994	50346900.42
BRAZIL	1998	27217924.38	GERMANY	1993	50991895.81
BRAZIL	1997	48378669.20	GERMANY	1992	48274126.10
BRAZIL	1996	50482870.36	INDIA	1998	29943144.35
BRAZIL	1995	47623383.63	INDIA	1997	50665453.23
BRAZIL	1994	47840165.73	INDIA	1996	50283092.29
BRAZIL	1993	49054694.04	INDIA	1995	50006774.64
BRAZIL	1992	48667639.08	INDIA	1994	48995190.76
CANADA	1998	30379833.77	INDIA	1993	50286902.85
CANADA	1997	50465052.31	INDIA	1992	50850329.40
CANADA	1996	52560501.39	INDONESIA	1998	27672340.00
CANADA	1995	52375332.81	INDONESIA	1997	50512145.73
CANADA	1994	52600364.66	INDONESIA	1996	51653060.12
CANADA	1993	52644504.07	INDONESIA	1995	51508779.59
CANADA	1992	53932871.70	INDONESIA	1994	52817950.32
CHINA	1998	31075466.16	INDONESIA	1993	47959994.96
CHINA	1997	50551874.45	INDONESIA	1992	51776605.03
CHINA	1996	51039293.88	IRAN	1998	29065736.24
CHINA	1995	49287534.62	IRAN	1997	50042063.05
CHINA	1994	50851090.07	IRAN	1996	50926653.19
CHINA	1993	54229629.83	IRAN	1995	51249667.65
CHINA	1992	52400529.37	IRAN	1994	50337085.87
EGYPT	1998	29054433.39	IRAN	1993	51730763.49
EGYPT	1997	50627611.45	IRAN	1992	49955856.56
EGYPT	1996	49542212.84	IRAQ	1998	31624551.00
EGYPT	1995	48311550.32	IRAQ	1997	55121749.02
EGYPT	1994	49790644.74	IRAQ	1996	55897663.79
EGYPT	1993	48904292.97	IRAQ	1995	54815472.52
EGYPT	1992	49434932.62	IRAQ	1994	54408516.13
ETHIOPIA	1998	28040717.27	IRAQ	1993	53633167.98
ETHIOPIA	1997	47455009.87	IRAQ	1992	55891939.34
ETHIOPIA	1996	46491097.57	JAPAN	1998	27934179.67
ETHIOPIA	1995	46804449.30	JAPAN	1997	44517162.55
ETHIOPIA	1994	48516143.92	JAPAN	1996	42545606.12
ETHIOPIA	1993	46551891.56	JAPAN	1995	43749356.40
ETHIOPIA	1992	44934648.64	JAPAN	1994	44840243.07
FRANCE	1998	32226407.84	JAPAN	1993	44660015.53
FRANCE	1997	47121485.86	JAPAN	1992	45410249.12
FRANCE	1996	47263135.50	JORDAN	1998	26901488.58
FRANCE	1995	47275997.57	JORDAN	1997	45471878.41
FRANCE	1994	47067209.33	JORDAN	1996	46794325.79
FRANCE	1993	51163370.11	JORDAN	1995	45178828.58
FRANCE	1992	47846235.33	JORDAN	1994	45333636.51
GERMANY	1998	28624942.66	JORDAN	1993	47971496.10

JORDAN	1992	44717239.18	SAUDI ARABIA	1994	51389601.97
KENYA	1998	28597614.34	SAUDI ARABIA	1993	52937508.88
KENYA	1997	47949733.73	SAUDI ARABIA	1992	54843459.64
KENYA	1996	46886924.62	UNITED KINGDOM	1998	28494874.00
KENYA	1995	46072338.76	UNITED KINGDOM	1997	49381810.90
KENYA	1994	45772061.17	UNITED KINGDOM	1996	51386853.96
KENYA	1993	46308728.23	UNITED KINGDOM	1995	51509586.79
KENYA	1992	47257780.84	UNITED KINGDOM	1994	48086499.71
MOROCCO	1998	26732115.58	UNITED KINGDOM	1993	49166827.22
MOROCCO	1997	45637304.25	UNITED KINGDOM	1992	49349122.08
MOROCCO	1996	45558221.75	UNITED STATES	1998	25126238.95
MOROCCO	1995	47851318.89	UNITED STATES	1997	50077306.42
MOROCCO	1994	46272172.94	UNITED STATES	1996	48048649.47
MOROCCO	1993	46764326.18	UNITED STATES	1995	48809032.42
MOROCCO	1992	48122783.58	UNITED STATES	1994	49296747.18
MOZAMBIQUE	1998	30712392.01	UNITED STATES	1993	48029946.80
MOZAMBIQUE	1997	50316528.76	UNITED STATES	1992	48671944.50
MOZAMBIQUE	1996	51640320.25	VIETNAM	1998	30442736.06
MOZAMBIQUE	1995	50693774.51	VIETNAM	1997	50309179.79
MOZAMBIQUE	1994	49253277.63	VIETNAM	1996	50488161.41
MOZAMBIQUE	1993	49153016.54	VIETNAM	1995	49658284.61
MOZAMBIQUE	1992	48247551.85	VIETNAM	1994	50596057.26
PERU	1998	29326102.32	VIETNAM	1993	50953919.15
PERU	1997	49753780.40	VIETNAM	1992	49613838.32
PERU	1996	50935170.29			
PERU	1995	53309883.41			
PERU	1994	50643531.80			
PERU	1993	51584622.00			
PERU	1992	47523899.05			
ROMANIA	1998	30368667.40			
ROMANIA	1997	50365683.85			
ROMANIA	1996	49598999.01			
ROMANIA	1995	47537642.87			
ROMANIA	1994	51455283.01			
ROMANIA	1993	50407136.89			
ROMANIA	1992	48185385.13			
RUSSIA	1998	28322384.03			
RUSSIA	1997	50106685.18			
RUSSIA	1996	51753342.43			
RUSSIA	1995	49215820.36			
RUSSIA	1994	52205666.44			
RUSSIA	1993	51860230.03			
RUSSIA	1992	53251677.15			
SAUDI ARABIA	1998	31541259.81			
SAUDI ARABIA	1997	52438750.81			
SAUDI ARABIA	1996	52543737.82			
SAUDI ARABIA	1995	52938696.53			

Submitting SQL Request #10:

```

/* Teradata
*/
/* @(#)TERADATA 10.sql 1.1.1.1@(#) */
/* Query 10 - Var_0 Rev_01 - TPC-H/TPC-R Returned Item Reporting
Query */

SELECT
    C_CUSTKEY,
    C_NAME,
    SUM(L_EXTENDEDPRI*(1-L_DISCOUNT)) (FLOAT)
(DECIMAL(18,2)) AS REVENUE,
    C_ACCTBAL,
    N_NAME,
    C_ADDRESS,
    C_PHONE,
    C_COMMENT
FROM
    CUSTOMER,
    ORDERTBL,
    LINEITEM,

```



```

NATION
WHERE
  C_CUSTKEY = O_CUSTKEY
  AND L_ORDERKEY = O_ORDERKEY
  AND O_ORDERDATE >= '1993-10-01'
  AND O_ORDERDATE < DATE '1993-10-01' + INTERVAL '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;

```

Query 10 complete, 37967 rows returned

only displaying 20 rows per TPC-H spec

Col1	Col2	Col3	Col4
57040	Customer#000057040	734235.25	632.8
7	JAPAN Eioyzjf4pp		3466
143347	Customer#000143347	721002.69	2557.4
7	EGYPT laReFYv,Kw4		3718
60838	Customer#000060838	679127.31	2454.7
7	BRAZIL 64EaJ5vMAHWJlBOxJklpNc2RjiWE		

```

9813|furiously even pinto beans integrate under the ruthless
foxes; ironic, even
dolphins across the slyl|
| 101998|Customer#000101998 | 637029.57|
3790.8
9|UNITED KINGDOM |01c9CILnNtfoQYmZj
|33-593-865-
6378|accounts doze blithely! enticing, final deposits sleep
blithely special acc
ounts. slyly express accounts pla|
| 125341|Customer#000125341 | 633508.09|
4983.5
1|GERMANY |S29ODD6bceU8QSuuEJznkNaK
|17-582-695-
5962|quickly express requests wake quickly blithely
|
| 25501|Customer#000025501 | 620269.78|
7725.0
4|ETHIOPIA | W556MXuoiaYCCZamJI,Rn0B4ACUGdkQ8DZ
|15-874-808-
6793|quickly special requests sleep evenly among the special
deposits. special d
eposi|
| 115831|Customer#000115831 | 596423.87|
5098.1
0|FRANCE |rFeBbEEyk dl
ne7zV5fDrmiq1oK09wV7pxqCgIc|16-715-386-
3788|carefully bold excuses sleep alongside of the thinly idle
|
| 84223|Customer#000084223 | 594998.02|
528.6
5|UNITED KINGDOM |nAVZCs6BaWap rrM27N 2qBnzc5WBauxbA
|33-442-824-
8191|pending, final ideas haggle final requests. unusual, regular
asymptotes aff
ix according to the even foxes. |
| 54289|Customer#000054289 | 585603.39|
5583.0
2|IRAN |vXCxoCsU0Bad5JQI ,oobkZ
|20-834-292-
4707|express requests sublute blithely regular requests. regular,
even ideas sol
ve. |
| 39922|Customer#000039922 | 584878.11|
7321.1
1|GERMANY |Zgy4s5012GKN4pLDPBU8m342gIw6R
|17-147-757-

```

8036|even pinto beans haggle. slyly bold accounts inte
| 6226|Customer#00006226 | 576783.76|
2230.0
9|UNITED KINGDOM |8gPu8,NPGkfyQQ0hcIYUGPIBwc,ybP5g,
|33-657-701-
3391|quickly final requests against the regular instructions wake
blithely final
instructions. pa|
| 922|Customer#00000922 | 576767.53|
3869.2
5|GERMANY |Az9RFaut7NkPnc5zSD2PwHgVwr4jRzq
|17-945-916-
9648|boldly final requests cajole blith
|
| 147946|Customer#000147946 | 576455.13|
2030.1
3|ALGERIA |iANyZHjqhy7Ajah0pTrYyhJ
|10-886-956-
3143|furiously even accounts are blithely above the furiousl
|
| 115640|Customer#000115640 | 569341.19|
6436.1
0|ARGENTINA |Vtgfia9qI 7EpHgecU1X
|11-411-543-
4901|final instructions are slyly according to the
|
| 73606|Customer#000073606 | 568656.86|
1785.6
7|JAPAN |xuR0Tro5yChDfOCrjkd2ol
|22-437-653-
6966|furiously bold orbits about the furiously busy requests wake
across the fur
iously quiet theodolites. d|
| 110246|Customer#000110246 | 566842.98|
7763.3
5|VIETNAM |7KzflgX MDOq7sOkI
|31-943-426-
9837|dolphins sleep blithely among the slyly final
|
| 142549|Customer#000142549 | 563537.24|
5085.9
9|INDONESIA |ChqEoK430ysjdHbtKCp6dKqjNyvvi9
|19-955-562-
2398|regular, unusual dependencies boost slyly; ironic
attainments nag fluffily
into the unusual packages?|

| 146149|Customer#000146149 | 557254.99|
1791.5
5|ROMANIA |s87fvzFQpU
|29-744-164-
6487|silent, unusual requests detect quickly slyly regul
|
| 52528|Customer#000052528 | 556397.35|
551.7
9|ARGENTINA |NFztyTOR10UOJ
|11-208-192-
3205|unusual requests detect. slyly dogged theodolites use slyly.
deposit
|
| 23431|Customer#000023431 | 554269.54|
3381.8
6|ROMANIA |HgiV0phqhaIa9aydNoIlb
|29-915-458-
2654|instructions nag quickly. furiously bold accounts cajol
|
Submitting SQL Request #11:

/* Teradata
*/
/* @(#)TERADATA 11.sql 1.1.1.1@(#) */
/* Query 11 - Var_0 Rev_01 - TPC-H/TPC-R Important Stock
Identification Query */

SELECT
PS_PARTKEY,
SUM(PS_SUPPLYCOST * PS_AVAILQTY (FLOAT)) (DEC(18,2)) AS
"VALUE"
FROM
PARTSUPP,
SUPPLIER,
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
(FLOAT)) * 0.000

1000000

```

FROM
    PARTSUPP,
    SUPPLIER,
    NATION
WHERE
    PS_SUPPKEY = S_SUPPKEY
    AND S_NATIONKEY = N_NATIONKEY
    AND N_NAME = 'GERMANY'

```

```

)
ORDER BY
    "VALUE" DESC;

```

Query 11 complete, 1048 rows returned

Col1	Col2		
129760	17538456.86	92406	13287414.50
166726	16503353.92	182636	13223726.74
191287	16474801.97	199969	13135288.21
161758	16101755.54	62865	13001926.94
34452	15983844.72	7284	12945298.19
139035	15907078.34	197867	12944510.52
9403	15451755.62	11562	12931575.51
154358	15212937.88	75165	12916918.12
38823	15064802.86	97175	12911283.50
85606	15053957.15	140840	12896562.23
33354	14408297.40	65241	12890600.46
154747	14407580.68	166120	12876927.22
82865	14235489.78	9035	12863828.70
76094	14094247.04	144616	12853549.30
222	13937777.74	176723	12832309.74
121271	13908336.00	170884	12792136.58
55221	13716120.47	29790	12723300.33
22819	13666434.28	95213	12555483.73
76281	13646853.68	183873	12550533.05
85298	13581154.93	171235	12476538.30
85158	13554904.00	21533	12437821.32
139684	13535538.72	17290	12432159.50
31034	13498025.25	156397	12260623.50
87305	13482847.04	122611	12222812.98
10181	13445148.75	139155	12220319.25
62323	13411824.30	146316	12215800.61
26489	13377256.38	171381	12199734.52
96493	13339057.83	198633	12078226.95
56548	13329014.97	167417	12046637.62
55576	13306843.35	59512	12043468.76
159751	13306614.48	31688	12034893.64
		159586	12001505.84
		8993	11963814.30
		120302	11857707.55
		43536	11779340.52
		9552	11776909.16
		86223	11772205.08
		53776	11758669.65
		131285	11616953.74
		91628	11611114.83
		169644	11567959.72
		182299	11567462.05
		33107	11453818.76
		104184	11436657.44
		67027	11419127.14
		176869	11371451.71
		30885	11369674.79

54420	11345076.88	58708	10466280.44
72240	11313951.05	89768	10465477.22
178708	11294635.17	146493	10444291.58
81298	11273686.13	55424	10444006.48
158324	11243442.72	16560	10425574.74
117095	11242535.24	133114	10415097.90
176793	11237733.38	195810	10413625.20
86091	11177793.79	76673	10391977.18
116033	11145434.36	97305	10390890.57
129058	11119112.20	134210	10387210.02
193714	11104706.39	188536	10386529.92
117195	11077217.96	122255	10335760.32
49851	11043701.78	2682	10312966.10
19791	11030662.62	43814	10303086.61
75800	11012401.62	34767	10290405.18
161562	10996371.69	165584	10273705.89
10119	10980015.75	2231	10270415.55
39185	10970042.56	111259	10263256.56
47223	10950022.13	195578	10239795.82
175594	10942923.05	21093	10217531.30
111295	10893675.61	29856	10216932.54
155446	10852764.57	133686	10213345.76
156391	10839810.38	87745	10185509.40
40884	10837234.19	135153	10179379.70
141288	10837130.21	11773	10167410.84
152388	10830977.82	76316	10165151.70
33449	10830858.72	123076	10161225.78
149035	10826130.02	91894	10130462.19
162620	10814275.68	39741	10128387.52
118324	10791788.10	111753	10119780.98
38932	10777541.75	142729	10104748.89
121294	10764225.22	116775	10097750.42
48721	10762582.49	102589	10034784.36
63342	10740132.60	186268	10012181.57
5614	10724668.80	44545	10000286.48
62266	10711143.10	23307	9966577.50
100202	10696675.55	124281	9930018.90
197741	10688560.72	69604	9925730.64
169178	10648522.80	21971	9908982.03
5271	10639392.65	58148	9895894.40
34499	10584177.10	16532	9886529.90
71108	10569117.56	159180	9883744.43
137132	10539880.47	74733	9877582.88
78451	10524873.24	35173	9858275.92
150827	10503810.48	7116	9856881.02
107237	10488030.84	124620	9838589.14
101727	10473558.10	122108	9829949.35

```

67200|          9828690.69|
164775|          9821424.44|
  9039|          9816447.72|
 14912|          9803102.20|
190906|          9791315.70|
130398|          9781674.27|
119310|          9776927.21|
  10132|          9770930.78|
107211|          9757586.25|
113958|          9757065.50|
  37009|          9748362.69|
  66746|          9743528.76|
134486|          9731922.00|
  15945|          9731096.45|
  55307|          9717745.80|
  56362|          9714922.83|
  57726|          9711792.10|
  57256|          9708621.00|
112292|          9701653.08|
  87514|          9699492.53|
174206|          9680562.02|
  72865|          9679043.34|
114357|          9671017.44|
112807|          9665019.21|
115203|          9661018.73|
177454|          9658906.35|
161275|          9634313.71|
  61893|          9617095.44|

```

[ROWS CUT]

Submitting SQL Request #12:

```

/* Teradata
*/
/* @(#)TERADATA 12.sql 1.1.1.1@(#) */
/* Query 12 - Var_0 Rev_01 - TPC-H/TPC-R Shipping Modes and Order
Priority Query
*/

```

```

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
  ORDERTBL,
  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 >= '1994-01-01'
  AND L_RECEIPTDATE < DATE '1994-01-01' + INTERVAL '1' YEAR
GROUP BY
  L_SHIPMODE
ORDER BY
  L_SHIPMODE;

```

Query 12 complete, 2 rows returned

Col1_____	_____Col2	_____Col3
MAIL	6202	9324
SHIP	6200	9262

Submitting SQL Request #1:

```

/* Teradata
*/
/* @(#)TERADATA 13.sql 1.1.1.1@(#) */
/* Query 13 - Var_0 Rev_01 - TPC-H/TPC-R Customer Distribution
Query */

```

```

SELECT
  C_COUNT, COUNT(*) AS CUSTDIST
FROM (
  SELECT
    C_CUSTKEY,
    COUNT(O_ORDERKEY)
  FROM
    CUSTOMER LEFT OUTER JOIN ORDERTBL 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;

```

Query 13 complete, 42 rows returned

Col1	Col2
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

Submitting SQL Request #14:

```

/* Teradata
*/
/* @(#)TERADATA 14.sql 1.1.1.1@(#) */
/* Query 14 - Var_0 Rev_01 - TPC-H/TPC-R Promotion Effect Query
*/

SELECT
    100.00*SUM(CASE
        WHEN P_TYPE LIKE 'PROMO%'
        THEN L_EXTENDEDPRIE*(1-L_DISCOUNT)
        ELSE 0
    END (FLOAT)) / SUM(L_EXTENDEDPRIE*(1-L_DISCOUNT)
(FLOAT)) (DECIMAL (18,
2)) AS PROMO_REVENUE
FROM
    LINEITEM,
    PARTBL
WHERE
    L_PARTKEY = P_PARTKEY
    AND L_SHIPDATE >= '1995-09-01'
    AND L_SHIPDATE < DATE '1995-09-01' + INTERVAL '1' MONTH;

```

Query 14 complete, 1 rows returned

Col1
16.38

Submitting SQL Request #15:

```

/* Teradata
*/
/* @(#)TERADATA 15.sql 1.1.1.1@(#) */
/* Query 15 - Var_0 Rev_01 - TPC-H/TPC-R Top Supplier Query */

CREATE VIEW REVENUE0 (SUPPLIER_NO,TOTAL_REVENUE) AS
SELECT
    L_SUPPKEY,

```

```

SUM(L_EXTENDEDPRI*(1-L_DISCOUNT) (FLOAT))
(DECEMAL(18,2))
FROM
LINEITEM
WHERE
L_SHIPDATE >= '1996-01-01'
AND L_SHIPDATE < DATE '1996-01-01' + INTERVAL '3'
MONTH
GROUP BY
L_SUPPKEY;

```

SQL statement complete.

```

SELECT
S_SUPPKEY,
S_NAME,
S_ADDRESS,
S_PHONE,
TOTAL_REVENUE
FROM
SUPPLIER,
REVENUE0
WHERE
S_SUPPKEY = SUPPLIER_NO
AND TOTAL_REVENUE = (
SELECT
MAX(TOTAL_REVENUE)
FROM
REVENUE0
)
ORDER BY
S_SUPPKEY;

```

Query 15 complete, 1 rows returned

Col1	Col2	Col3
8449	Supplier#000008449	Wp34zim9qYFbVctdW
20-469-856-8873	1772627.21	

DROP VIEW REVENUE0;

SQL statement complete.

Submitting SQL Request #16:

```

/* Teradata
*/
/* @(#)TERADATA 16.sql 1.1.1.1@(#) */
/* Query 16 - Var_0 Rev_01 - TPC-H/TPC-R Parts/Supplier
Relationship Query */

```

```

SELECT
P_BRAND,
P_TYPE,
P_SIZE,
COUNT(DISTINCT PS_SUPPKEY) AS SUPPLIER_CNT
FROM
PARTSUPP,
PARTTBL
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
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;

```

Query 16 complete, 18314 rows returned

Col1	Col2	Col3	Col4
Brand#41	MEDIUM BRUSHED TIN	3	28
Brand#54	STANDARD BRUSHED COPPER	14	27
Brand#11	STANDARD BRUSHED TIN	23	24
Brand#11	STANDARD BURNISHED BRASS	36	24
Brand#15	MEDIUM ANODIZED NICKEL	3	24
Brand#15	SMALL ANODIZED BRASS	45	24

Brand#15	SMALL BURNISHED NICKEL	19	24	Brand#21	LARGE ANODIZED COPPER	36	20
Brand#21	MEDIUM ANODIZED COPPER	3	24	Brand#21	LARGE BRUSHED TIN	3	20
Brand#22	SMALL BRUSHED NICKEL	3	24	Brand#21	MEDIUM ANODIZED COPPER	14	20
Brand#22	SMALL BURNISHED BRASS	19	24	Brand#21	PROMO BRUSHED TIN	36	20
Brand#25	MEDIUM BURNISHED COPPER	36	24	Brand#21	PROMO POLISHED NICKEL	45	20
Brand#31	PROMO POLISHED COPPER	36	24	Brand#21	SMALL ANODIZED COPPER	9	20
Brand#33	LARGE POLISHED TIN	23	24	Brand#21	SMALL POLISHED NICKEL	23	20
Brand#33	PROMO POLISHED STEEL	14	24	Brand#22	LARGE ANODIZED COPPER	36	20
Brand#35	PROMO BRUSHED NICKEL	14	24	Brand#22	LARGE BRUSHED COPPER	49	20
Brand#41	ECONOMY BRUSHED STEEL	9	24	Brand#22	PROMO ANODIZED TIN	49	20
Brand#41	ECONOMY POLISHED TIN	19	24	Brand#22	PROMO POLISHED BRASS	45	20
Brand#41	LARGE PLATED COPPER	36	24	Brand#22	SMALL BURNISHED STEEL	45	20
Brand#42	ECONOMY PLATED BRASS	3	24	Brand#23	MEDIUM ANODIZED STEEL	45	20
Brand#42	STANDARD POLISHED TIN	49	24	Brand#23	PROMO POLISHED STEEL	23	20
Brand#43	PROMO BRUSHED TIN	3	24	Brand#23	STANDARD BRUSHED TIN	14	20
Brand#43	SMALL ANODIZED COPPER	36	24	Brand#23	STANDARD PLATED NICKEL	36	20
Brand#44	STANDARD POLISHED NICKEL	3	24	Brand#24	PROMO PLATED COPPER	49	20
Brand#52	ECONOMY PLATED TIN	14	24	Brand#24	PROMO PLATED STEEL	49	20
Brand#52	STANDARD BURNISHED NICKEL	3	24	Brand#24	PROMO POLISHED STEEL	9	20
Brand#53	MEDIUM ANODIZED STEEL	14	24	Brand#24	STANDARD BRUSHED TIN	36	20
Brand#14	PROMO ANODIZED NICKEL	45	23	Brand#25	LARGE ANODIZED BRASS	3	20
Brand#32	ECONOMY PLATED BRASS	9	23	Brand#25	PROMO BURNISHED TIN	3	20
Brand#52	SMALL ANODIZED COPPER	3	23	Brand#31	ECONOMY POLISHED NICKEL	3	20
Brand#11	ECONOMY BRUSHED COPPER	45	20	Brand#31	MEDIUM PLATED TIN	45	20
Brand#11	ECONOMY PLATED BRASS	23	20	Brand#31	SMALL ANODIZED STEEL	14	20
Brand#11	LARGE BRUSHED COPPER	49	20	Brand#32	ECONOMY ANODIZED COPPER	36	20
Brand#11	LARGE POLISHED COPPER	49	20	Brand#32	ECONOMY BRUSHED NICKEL	49	20
Brand#12	STANDARD ANODIZED TIN	49	20	Brand#32	LARGE ANODIZED TIN	19	20
Brand#12	STANDARD PLATED BRASS	19	20	Brand#32	MEDIUM BURNISHED COPPER	19	20
Brand#13	ECONOMY BRUSHED BRASS	9	20	Brand#32	SMALL ANODIZED STEEL	45	20
Brand#13	ECONOMY BURNISHED STEEL	14	20	Brand#33	ECONOMY POLISHED COPPER	19	20
Brand#13	LARGE BURNISHED NICKEL	19	20	Brand#33	PROMO PLATED NICKEL	14	20
Brand#13	MEDIUM BURNISHED COPPER	36	20	Brand#33	SMALL POLISHED TIN	9	20
Brand#13	SMALL BRUSHED TIN	45	20	Brand#33	STANDARD ANODIZED BRASS	49	20
Brand#13	STANDARD ANODIZED COPPER	3	20	Brand#33	STANDARD BURNISHED BRASS	45	20
Brand#13	STANDARD PLATED NICKEL	23	20	Brand#34	ECONOMY BRUSHED NICKEL	49	20
Brand#14	ECONOMY ANODIZED COPPER	14	20	Brand#34	LARGE BRUSHED BRASS	19	20
Brand#14	ECONOMY PLATED TIN	36	20	Brand#34	SMALL BRUSHED TIN	3	20
Brand#14	ECONOMY POLISHED NICKEL	3	20	Brand#34	STANDARD PLATED COPPER	9	20
Brand#14	MEDIUM ANODIZED NICKEL	3	20	Brand#35	LARGE ANODIZED NICKEL	3	20
Brand#14	SMALL POLISHED TIN	14	20	Brand#35	MEDIUM ANODIZED BRASS	45	20
Brand#15	MEDIUM ANODIZED COPPER	9	20	Brand#35	MEDIUM ANODIZED STEEL	23	20
Brand#15	MEDIUM PLATED TIN	23	20	Brand#35	PROMO ANODIZED COPPER	49	20
Brand#15	PROMO PLATED BRASS	14	20	Brand#35	SMALL POLISHED COPPER	14	20
Brand#15	SMALL ANODIZED COPPER	45	20	Brand#41	LARGE ANODIZED STEEL	3	20
Brand#15	SMALL PLATED COPPER	49	20	Brand#41	LARGE BRUSHED NICKEL	23	20
Brand#15	STANDARD PLATED TIN	3	20	Brand#41	LARGE BURNISHED COPPER	3	20

Brand#41	MEDIUM PLATED STEEL	19	20	Brand#11	LARGE ANODIZED COPPER	14	16
Brand#41	SMALL BURNISHED COPPER	23	20	Brand#11	LARGE BRUSHED TIN	45	16
Brand#42	MEDIUM BURNISHED BRASS	14	20	Brand#11	LARGE BURNISHED COPPER	23	16
Brand#42	SMALL BURNISHED COPPER	3	20	Brand#11	LARGE BURNISHED NICKEL	36	16
Brand#43	ECONOMY POLISHED COPPER	9	20	Brand#11	LARGE PLATED STEEL	14	16
Brand#43	SMALL PLATED STEEL	3	20	Brand#11	MEDIUM BRUSHED NICKEL	14	16
Brand#43	STANDARD BURNISHED TIN	23	20	Brand#11	MEDIUM BRUSHED STEEL	49	16
Brand#44	LARGE ANODIZED STEEL	23	20	Brand#11	MEDIUM BURNISHED NICKEL	49	16
Brand#44	PROMO ANODIZED TIN	23	20	Brand#11	MEDIUM BURNISHED TIN	3	16
Brand#51	ECONOMY BRUSHED BRASS	49	20	Brand#11	MEDIUM PLATED COPPER	9	16
Brand#51	ECONOMY POLISHED NICKEL	9	20	Brand#11	PROMO ANODIZED BRASS	19	16
Brand#51	MEDIUM BRUSHED TIN	9	20	Brand#11	PROMO ANODIZED BRASS	49	16
Brand#51	MEDIUM PLATED BRASS	9	20	Brand#11	PROMO ANODIZED STEEL	45	16
Brand#51	PROMO BURNISHED BRASS	9	20	Brand#11	PROMO PLATED BRASS	45	16
Brand#51	SMALL PLATED NICKEL	49	20	Brand#11	SMALL ANODIZED TIN	45	16
Brand#51	STANDARD ANODIZED NICKEL	49	20	Brand#11	SMALL BRUSHED STEEL	49	16
Brand#51	STANDARD BRUSHED COPPER	3	20	Brand#11	SMALL BURNISHED COPPER	19	16
Brand#52	ECONOMY ANODIZED BRASS	3	20	Brand#11	SMALL BURNISHED COPPER	45	16
Brand#52	ECONOMY BRUSHED COPPER	49	20	Brand#11	SMALL BURNISHED NICKEL	14	16
Brand#52	LARGE ANODIZED NICKEL	45	20	Brand#11	SMALL POLISHED NICKEL	36	16
Brand#52	MEDIUM ANODIZED TIN	23	20	Brand#11	STANDARD ANODIZED BRASS	19	16
Brand#52	MEDIUM BURNISHED TIN	45	20	Brand#11	STANDARD ANODIZED COPPER	14	16
Brand#52	SMALL PLATED COPPER	36	20	Brand#11	STANDARD BRUSHED STEEL	45	16
Brand#52	STANDARD ANODIZED BRASS	45	20	Brand#11	STANDARD POLISHED NICKEL	23	16
Brand#53	ECONOMY PLATED COPPER	45	20	Brand#12	ECONOMY ANODIZED TIN	14	16
Brand#53	PROMO ANODIZED COPPER	49	20	Brand#12	ECONOMY BRUSHED COPPER	9	16
Brand#53	PROMO BRUSHED COPPER	23	20	Brand#12	ECONOMY BRUSHED COPPER	36	16
Brand#53	PROMO PLATED TIN	19	20	Brand#12	ECONOMY BURNISHED BRASS	9	16
Brand#53	PROMO POLISHED NICKEL	3	20	Brand#12	ECONOMY BURNISHED NICKEL	36	16
Brand#53	SMALL ANODIZED STEEL	9	20	Brand#12	LARGE ANODIZED BRASS	14	16
Brand#53	SMALL BRUSHED COPPER	3	20	Brand#12	LARGE ANODIZED COPPER	9	16
Brand#53	SMALL BRUSHED NICKEL	3	20	Brand#12	LARGE ANODIZED STEEL	23	16
Brand#54	ECONOMY PLATED STEEL	9	20	Brand#12	LARGE BURNISHED TIN	36	16
Brand#54	ECONOMY POLISHED TIN	3	20	Brand#12	LARGE PLATED COPPER	49	16
Brand#54	SMALL BRUSHED BRASS	19	20	Brand#12	LARGE POLISHED COPPER	49	16
Brand#55	MEDIUM ANODIZED COPPER	3	20	Brand#12	MEDIUM PLATED COPPER	19	16
Brand#55	PROMO BURNISHED STEEL	14	20	Brand#12	MEDIUM PLATED NICKEL	23	16
Brand#55	PROMO POLISHED NICKEL	49	20	Brand#12	PROMO ANODIZED BRASS	45	16
Brand#55	STANDARD ANODIZED BRASS	19	20	Brand#12	PROMO ANODIZED STEEL	49	16
Brand#55	STANDARD BURNISHED COPPER	45	20	Brand#12	PROMO BURNISHED STEEL	9	16
Brand#43	ECONOMY ANODIZED TIN	3	19	Brand#12	SMALL BRUSHED NICKEL	36	16
Brand#11	ECONOMY ANODIZED BRASS	14	16	Brand#12	SMALL BRUSHED TIN	45	16
Brand#11	ECONOMY ANODIZED BRASS	23	16	Brand#12	STANDARD ANODIZED BRASS	3	16
Brand#11	ECONOMY ANODIZED COPPER	14	16	Brand#12	STANDARD ANODIZED NICKEL	14	16
Brand#11	ECONOMY BRUSHED BRASS	49	16	Brand#12	STANDARD BRUSHED BRASS	3	16
Brand#11	ECONOMY BRUSHED STEEL	19	16	Brand#12	STANDARD BRUSHED TIN	9	16
Brand#11	ECONOMY BURNISHED NICKEL	23	16	Brand#12	STANDARD BRUSHED TIN	36	16

Brand#12	STANDARD POLISHED COPPER	9	16
Brand#13	ECONOMY ANODIZED STEEL	45	16
Brand#13	ECONOMY POLISHED BRASS	3	16
Brand#13	LARGE BRUSHED NICKEL	23	16
Brand#13	LARGE BURNISHED NICKEL	9	16
Brand#13	MEDIUM BRUSHED STEEL	49	16

[ROWS CUT]

Submitting SQL Request #17:

```

/* Teradata
*/
/* @(#)TERADATA 17.sql 1.1.1.1@(#) */
/* Query 17 - Var_0 Rev_01 - TPC-H/TPC-R Small-Quantity-Order
Revenue Query */

```

```

SELECT
  SUM(L_EXTENDEDPRI) / 7.0 AS AVG_YEARLY
FROM
  LINEITEM,
  PARTBL
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
      LINEITEM
    WHERE
      L_PARTKEY = P_PARTKEY
  );

```

Query 17 complete, 1 rows returned

_____Col1
348406.05

Submitting SQL Request #18:

```

/* Teradata
*/
/* @(#)TERADATA 18.sql 1.1.1.1@(#) */
/* Query 18 - Var_0 Rev_01 - TPC-H/TPC-R Large Volume Customer
Query */

```

```

/* Return the first 100 selected rows
*/

```

```

SELECT
  C_NAME,
  C_CUSTKEY,
  O_ORDERKEY,
  O_ORDERDATE,
  O_TOTALPRICE,
  SUM(L_QUANTITY (FLOAT)) (DECIMAL(18,2)) AS SUM_QTY
FROM
  CUSTOMER,
  ORDERTBL,
  LINEITEM
WHERE
  O_ORDERKEY IN (
    SELECT
      L_ORDERKEY
    FROM
      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;

```

Query 18 complete, 57 rows returned

Col1_____	_____Col2	_____Col3 _____	_____
Col4 _____	_____Col5 _____	_____Col6	_____
Customer#000128120	128120	4722021.00	1994-04-07
544089.09	323.00		

Customer#000144617 02-12	144617	3043270.00 1997-	469692.58	307.00	
530604.44	317.00		Customer#000148885 05-31	148885	2942469.00 1992-
Customer#000013940 04-13	13940	2232932.00 1997-	469630.44	313.00	
522720.61	304.00		Customer#000114586 05-19	114586	551136.00 1993-
Customer#000066790 09-30	66790	2199712.00 1996-	469605.59	308.00	
515531.82	327.00		Customer#000105260 09-06	105260	5296167.00 1996-
Customer#000046435 07-03	46435	4745607.00 1997-	469360.57	303.00	
508047.99	309.00		Customer#000147197 02-02	147197	1263015.00 1997-
Customer#000015272 07-28	15272	3883783.00 1993-	467149.67	320.00	
500241.33	302.00		Customer#000064483 07-04	64483	2745894.00 1996-
Customer#000146608 06-12	146608	3342468.00 1994-	466991.35	304.00	
499794.58	303.00		Customer#000136573 05-31	136573	2761378.00 1996-
Customer#000096103 03-16	96103	5984582.00 1992-	461282.73	301.00	
494398.79	312.00		Customer#000016384 04-12	16384	502886.00 1994-
Customer#000024341 11-15	24341	1474818.00 1992-	458378.92	312.00	
491348.26	302.00		Customer#000117919 06-20	117919	2869152.00 1996-
Customer#000137446 05-23	137446	5489475.00 1997-	456815.92	317.00	
487763.25	311.00		Customer#000012251 11-24	12251	735366.00 1993-
Customer#000107590 11-04	107590	4267751.00 1994-	455107.26	309.00	
485141.38	301.00		Customer#000120098 06-14	120098	1971680.00 1995-
Customer#000050008 12-09	50008	2366755.00 1996-	453451.23	308.00	
483891.26	302.00		Customer#000066098 08-07	66098	5007490.00 1992-
Customer#000015619 08-07	15619	3767271.00 1996-	453436.16	304.00	
480083.96	318.00		Customer#000117076 02-05	117076	4290656.00 1997-
Customer#000077260 09-12	77260	1436544.00 1992-	449545.85	301.00	
479499.43	307.00		Customer#000129379 06-07	129379	4720454.00 1997-
Customer#000109379 10-10	109379	5746311.00 1996-	448665.79	303.00	
478064.11	302.00		Customer#000126865 11-07	126865	4702759.00 1994-
Customer#000054602 02-09	54602	5832321.00 1997-	447606.65	320.00	
471220.08	307.00		Customer#000088876 12-30	88876	983201.00 1993-
Customer#000105995 07-03	105995	2096705.00 1994-	446717.46	304.00	

Customer#000036619 01-17	36619	4806726.00 1995-	418161.49	305.00	Customer#000012599 02-12	12599	4259524.00 1998-
446704.09	328.00		415200.61	304.00	Customer#000105410 03-05	105410	4478371.00 1996-
Customer#000141823 12-29	141823	2806245.00 1996-	Customer#000149842 05-30	302.00	149842		5156581.00 1994-
446269.12	310.00	2662214.00 1993-	411329.35	302.00	Customer#000010129 03-21	10129	5849444.00 1994-
Customer#000053029 08-13	53029		Customer#000069904 10-19	309.00	69904		1742403.00 1996-
446144.49	302.00	3037414.00 1995-	408513.00	305.00	Customer#000017746 04-09	17746	6882.00 1997-
Customer#000018188 01-25	18188		Customer#000013072 03-15	303.00	13072		1481925.00 1998-
443807.22	308.00	29158.00 1995-	399195.47	301.00	Customer#000082441 02-07	82441	857959.00 1994-
Customer#000066533 10-21	66533	4134341.00 1995-	Customer#000088703 01-30	305.00	88703		2995076.00 1994-
443576.50	305.00	4134341.00 1995-	363812.12	302.00			
Customer#000037729 06-29	37729		Submitting SQL Request #19:				
441082.97	309.00	2329187.00 1998-	/* Teradata				
Customer#000003566 01-04	3566		*/				
439803.36	304.00	4527553.00 1994-	/* @(#)TERADATA 19.sql 1.1.1.1@(#) */				
Customer#000045538 05-22	45538	4739650.00 1995-	/* Query 19 - Var_0 Rev_1 - TPC-H/TPC-R Discounted Revenue Query				
436275.31	305.00	4739650.00 1995-	*/				
Customer#000081581 11-04	81581		SELECT	SUM(L_EXTENDEDPRICE* (1 - L_DISCOUNT) (FLOAT))			
435405.90	305.00	1544643.00 1997-		(DECIMAL(18,2)) AS REVENU			
Customer#000119989 09-20	119989		E	FROM			
434568.25	320.00	3861123.00 1998-		LINEITEM,			
Customer#000003680 07-03	3680			PARTTBL			
433525.97	301.00	967334.00 1995-	WHERE	(
Customer#000113131 12-15	113131			P_PARTKEY = L_PARTKEY			
432957.75	301.00	565574.00 1995-					
Customer#000141098 09-24	141098						
430986.69	301.00	5200102.00 1997-					
Customer#000093392 01-22	93392						
425487.51	304.00	1845057.00 1994-					
Customer#000015631 05-12	15631						
419879.59	302.00	4439686.00 1996-					
Customer#000112987 09-17	112987						

```

        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'
    );

```

Query 19 complete, 1 rows returned

```

|_____Col1|
|          3083843.06|

```

Submitting SQL Request #20:

```

/* Teradata
*/
/* @(#)TERADATA 20.sql 1.1.1.1@(#) */
/* Query 20 - Var_0 Rev_01 - TPC-H/TPC-R The Potential Part
Promotion query */

```

```

SELECT
    S_NAME,

```

```

    S_ADDRESS
FROM
    SUPPLIER,
    NATION
WHERE
    S_SUPPKEY IN (
        SELECT
            PS_SUPPKEY
        FROM
            PARTSUPP
        WHERE
            PS_PARTKEY IN (
                SELECT
                    P_PARTKEY
                FROM
                    PARTTBL
                WHERE
                    P_NAME LIKE 'forest%'
            )
        AND PS_AVAILQTY > (
            SELECT
                0.5 * SUM(L_QUANTITY)
            FROM
                LINEITEM
            WHERE
                L_PARTKEY = PS_PARTKEY
                AND L_SUPPKEY = PS_SUPPKEY
                AND L_SHIPDATE >= '1994-01-01'
                AND L_SHIPDATE < DATE '1994-01-
01' + INTERVAL
                '1' YEAR
        )
        AND S_NATIONKEY = N_NATIONKEY
        AND N_NAME = 'CANADA'
ORDER BY
    S_NAME;

```

Query 20 complete, 204 rows returned

```

|Col1_____|Col2_____
|_____|
|Supplier#00000020      |iybAE,RmTymrZVYaFZva2SH,j
|
|Supplier#00000091      |YV45D7TkfdQanOOZ7q9QxkyGUapU1oOWU6q3
|

```

Supplier#00000197	YC2Acon6kjY3zj3Fbxs2k4Vdf7X0cd2F	Supplier#000001700	7hM1Cof1Y5zLFg
Supplier#00000226	83qOdU2EYRdPQAQhEtn GRZEd	Supplier#000001726	TeRY7TtTH24sEword7yAaSkjx8
Supplier#00000285	Br7elnntlyxrw6ImgpJ7YdhFDjuBf	Supplier#000001730	Rc8e,1Pybn r6zo0VJIEiD0UD vhk
Supplier#00000378	FfbhyCxWvcPr08ltp9	Supplier#000001746	qWsendlOekQG1aW4uq06uQaCm51se8lirv7
Supplier#00000402	i9Sw4DoyMhzhKXCH9By,AYSgmD	Supplier#000001752	Fra7outx41THYJaRThdOGiBk
Supplier#00000530	0qwCMwobKY OcmLyfRXlagA8ukENJv,	Supplier#000001856	jXcRgzYF0ah05iR8p6w5SbJLcUGyYiURPvFwUWM
Supplier#00000688	D fw5ocppmZpYBBIPi718hCihLDZ5KhKX	Supplier#000001931	FpJbMU2h6ZR2eBv8I9NIXF
Supplier#00000710	f19YPvOyb QoYwJKC,oPycpGfieBAcwKJo	Supplier#000001939	Nrk,JA4bfReUs
Supplier#00000736	l6i2nMwVuovfKnuVgaSGK2rDy65DlAFLegiL7	Supplier#000001990	DSDJkCgBJzuPglyuM,CUDlnsRliOxkkHezTCA
Supplier#00000761	z1SLelQUj2XrvTTFnv7WAcYZGvMTx882d4	Supplier#000002020	jB6r1d7MxP6co
Supplier#00000884	bmhEShejaS	Supplier#000002022	dwebGX7Id2pc25YvY33
Supplier#00000887	urEaTejH5POADP2ARrf	Supplier#000002036	20ytTtVObjKUUI2WCB0A
Supplier#00000935	ij98czM 2KzWe7dTOxB8sq0UfCdvrX	Supplier#000002204	uYmlr46C06udCqanj0KiRsoTQakZsEyssL
Supplier#00000975	,AC e,tBpNwKb5xMUzeohxlRn,	Supplier#000002243	nSOEV3JeOU79
hdZJo73gFQF8y		Supplier#000002245	hz2qWXWVjOyKhqPYMoEwz6zFkrTaDM
Supplier#000001263	rQWr6nf8ZhB2TAiIDivo5Io	Supplier#000002282	ES21K9dxoW1I1TzWCj7ekdlNwSWnv1Z
Supplier#000001399	LmrocnIMSyYOWuANx7	6mQ,BKn	
Supplier#000001446	lch9HMNU1R7a0LIybsUodVknk6	Supplier#000002303	nCoWfpB6YOymbgOht7ltfklpkH1
Supplier#000001454	TOpimgu2TVXIjhiL93h,	Supplier#000002373	RzHSxOTQmElCjxIBiVA52Z JB58rJhPRy1R
Supplier#000001500	wDmF5xLxtQch9ctVu,	Supplier#000002419	qydBQd14I515mVXa4fYY
Supplier#000001602	uKNWIEafaM644	Supplier#000002481	nLKHUOn2M19TOA06Znq9GEMcILMO2
Supplier#000001626	UhxNRzUuldtFmp0	Supplier#000002571	JZUugz04c iJFLrlGsz90 N,W 1rVHNIReyq
Supplier#000001682	pXTkGxrTQVyH1Rr	Supplier#000002585	CsPoKpw2QuTY4AV1NkWuttneIa4SN
Supplier#000001699	Q9C4rfJ26oijVPqqcqVXeRI	Supplier#000002630	ZIQAvjNuy9KH5ive zm7k V1PiDl7CCo21

Supplier#000002719	4nnzQI2CbqREQUuIsXTBVUkaP4mNS3	Supplier#000003421	Sh3dt9W5oeofFWovnFhrg,
Supplier#000002721	HVdFAN2JHMQSpKm	Supplier#000003441	zvFJIzS,oUuShHjpcX
Supplier#000002730	lIFxR4fzm3lC6,muzJwl84z	Supplier#000003590	sy79CMLxqb,Cbo
Supplier#000002775	yDclaDaBD4ihH	Supplier#000003607	lNqFHQYjwSAkf
Supplier#000002853	rTNAOIItXka	Supplier#000003625	qY588W0Yk5iaUy1RXTgNrEKrMAjBYHcKs
Supplier#000002875	6JgMi 9Qt6VmwL3Ltt1SRlKww0keLQ,RAZa	Supplier#000003656	eEYmmO2gmD JdfG32XtDgJV,db56
Supplier#000002934	m,trBENywsArwg3DhB	Supplier#000003782	iVsPZg7bk06TqNMwi0LKbLUrClzmrq
Supplier#000002941	Naddba 8YTEKekZyP0	Supplier#000003918	meRvRCsJoAbfqd0Re4
Supplier#000002960	KCPCEsRGG06vx8TygHh60nAYf9rStQT2T	Supplier#000003941	Pmb05mQfBMS61807WKqZJ 9vyv
Supplier#000002980	B9k9yVsyaXvWktOSHezqHiAEp9id0SKzkw	Supplier#000003994	W00LZp3NjK0
Supplier#000003062	LSQNgqYlxnOzz9zBCapy7HwOZQ	Supplier#000004005	V723F1wCy2eA40gIu8TjBtOVUHp
Supplier#000003087	ANwe8QsZ4rgjlHSqVz991eWQ	Supplier#000004033	ncsAhv9Je,kFXTNjfb2
Supplier#000003089	s5b VCIzqMSZVa r g7LTdgc29GbTE7rI1x	Supplier#000004140	0hL7DJyYjcHL
Supplier#000003095	HxON3jJhUi3zjt,r mTD	Supplier#000004165	wTJ2dZnQA8P2oi99N6DT47ndHy,XKD2
Supplier#000003201	E87yws6I,t0qNs4QW7UzExKiJnJDZWue	Supplier#000004207	tF64pwiOM4IkWjN3mS,e06WuAjLx
Supplier#000003213	pxrRP4irQ1VoyfQ,dTf3	Supplier#000004236	dl,HptJmGipxYsSqn9wmqkuWjst,mCeJ806T
Supplier#000003241	j06SU,LS903mwjAMOVIANeIhb	Supplier#000004246	Xha aXQF7u4qU3LsHD
Supplier#000003275	9x04nyJ2QJcX6vGf	Supplier#000004278	bBddbpbXIVp Di9
Supplier#000003288	EDdfNt7E5Uc,xLTupoIgyL4y7ujh,	Supplier#000004343	GK3sbopqrQEkwLMvVBFCG
Supplier#000003313	El2I7we,049SPrvomUm4hZwJoOhZkvLxLJXgVH	Supplier#000004346	S3076LEOwo
Supplier#000003314	jnisU8MzqO4iUB3zsPcrysMw3DDUojs4q7LD	Supplier#000004388	VfZ 1lJ,mwp4aS
Supplier#000003380	jPv0V,pszouuFT3YsAq1P,kxT3u,gTFiEbRt,x	Supplier#000004406	Ah0ZaLu6VwufPWUz,7kbXgYZhauEaHqGIg
Supplier#000003403	e3X2o ,KCG9tsHji8A XXCxiF2hZWBw	Supplier#000004430	yvSsKNSTL5HLXBET4luOsPNLxKzAMk

Supplier#000004522	xXtCKwsZDARxIBGdfzX2PgobGZsBg	Supplier#000005730	5rkb0PSews HvxkL8JaD41UpnSF2cg8H1
Supplier#000004527	p pVXCnxgcklWF6Alo3OHY3qW6	Supplier#000005736	2dq XTYhtYWSfp
Supplier#000004542	NJSbLJDroYG2y1r3rDiKg	Supplier#000005737	dmEWcS32C3kx,d,B95 OmYn48
Supplier#000004574	lHvGwnVueZ5CIndc	Supplier#000005797	,o,OebwRbSDmVl9gN9fpWPCiqB UogvlSR
Supplier#000004655	67NqBc4 t3PG3F8aO IsqWNq4kGaPowYL	Supplier#000005836	tx3SjPD2ZuWGFBRH,
Supplier#000004701	6jX4u47URzIMHF	Supplier#000005875	lK,sYiGzB94hSyHy9xvSZFbVQNCZe2LXZuGbS
Supplier#000004711	bEzjplQdQu ls2ERMxv0km vn6bu2zXlLl	Supplier#000005974	REhR5jE,lLusQXvf54SwYySgsSSVFhu
Supplier#000004987	UFx1upJ8MvOvgFjA8	Supplier#000005989	rjFY,5kgLpBu7c
Supplier#000005000	DeX804 w0H8FrCUvahgy ilbuzBX3NK	Supplier#000006059	4m0cv8MwJ9yX2vlwI Z
Supplier#000005100	OfvYPs3Io,wEvvLHNaLuCX	Supplier#000006065	UiI2Cy3W4Tu5sLk LuvXLRy6KihlGv
Supplier#000005192	JDp4rhXiDw0kf6RH	Supplier#000006070	TalC5m0pDr06DZbngfmGmqe
Supplier#000005195	Woi3b2ZaicPh ZSfulEfXhE	Supplier#000006109	rY5gbfh3dKHnlycQUTPGCwnbe
Supplier#000005283	5fxYXxwXy,TQX,MqDC2hxzyQ	Supplier#000006121	S92ycWwEzYYw4GspCBJN1WMuHhoZ
Supplier#000005300	gXG28YqpxU	Supplier#000006215	j2iEbTsl,5PwDqWZ7klyiISb7qtiizljdIPEo
Supplier#000005386	Ub6AAfHpWLWP	Supplier#000006217	RVN23SYT9jenUeaWGxUd
Supplier#000005426	9Dz2OVT1q sb4BK71ljQ1XjPBYRPvO	Supplier#000006274	S3yTZWqxTKUq g QQgcW9 AqhCkNZsW51hHuwU
Supplier#000005484	saFdOR	Supplier#000006435	xIge69XszYbn04Eon7cHH08y
qW7AFY,3asPqiiAa1lMo22pCoN0BtPrKo		Supplier#000006463	7 wkdj2EO49iotley2kmIM ADpLSszGV3RNWj
Supplier#000005505	d2sbjG43KwMPX	Supplier#000006493	ojV f,sNaB6Hm7r,fknDVTL63raJgAjZK
Supplier#000005506	On f5ypzoWgB	Supplier#000006521	b9 2zjHzxR
Supplier#000005516	XsN99Ks9wEvcohU6jRD2MeebQFf76mD8vovuy	Supplier#000006607	3F 2e2gqD5u5B
Supplier#000005536	Nzo9tGkpgbHT,EZ4D,77MYKl4ah1C	Supplier#000006706	Ak4ga,ePu1QZ6C3qkrqjosaX0gxvqS9vkbe
Supplier#000005605	7Vj6Eil0mThqkM	Supplier#000006761	n4jhxGMqB5prD1HhpLvwrWStOLlla
Supplier#000005631	l4TVrjlzo2SJEbYCDgpMwTlvsqC		

Supplier#000006808	HGd2Xo 9nEcHJhZvXjXxWKIpApT	Supplier#000007926	ErzCF80K9Uy
Supplier#000006858	fnlINT885vBBhsWwTGiZ0o22thwGY16h	Supplier#000007957	ELwnio14ssoU1 dRyZIL OK3Vtzb
GHJj21		Supplier#000007965	F7Un5lJ7p5hhj
Supplier#000006872	XIDPiA7PLXCWK6SeEcld	Supplier#000007968	DsF9UlZ2Fo6HXN9aErvyglikHoD582HSGZpP
Supplier#000006949	mLxYUJhsGcLtKe ,GFirNu183AvT	Supplier#000007998	LnASFBfYRF0o9d6d,asBvVq9Lo2P
Supplier#000006985	PrUUiboQpy,OtgJ0lZ4BxJQUyrw9c3I	Supplier#000008168	aOa82a8ZbKcfnDLX
Supplier#000007072	2tRyX9M1a 4Rcm57s779F1ANG9jlpK	Supplier#000008231	IK7eGw Yj90sTdpSP,vcqWxLB
Supplier#000007098	G3j8g0KC4OcbAu20VoPHrXQWMCUdjg8wgCHOExu	Supplier#000008243	2AyePMkDqzmzVzjGTizXthFL08h
Supplier#000007135	ls DoKV7V5ulfQy9V	EiudCMxOmIIG	Supplier#000008275
Supplier#000007160	TqDGBULB3cTqIT6FKDvm9BS4e4v,zwYiQPb	Supplier#000008323	75I18sZmASwm
Supplier#000007169	tEc95D2moN9S84nd550,dlnW	POeherMdj9tmpyeQ,BfCXN5BIAb	Supplier#000008366
Supplier#000007322	wr7dgte5q MAjiY0uwm3MyDkSMX1	h778cEj14BuW9OEKlvPTWq4iwASR6EBBXN7zeS8	Supplier#000008423
Supplier#000007365	51xhROLvQMJ05DndtZwt	RQhKnkAhR0DAR3Ix4Q1weMMn00hNe Kq	Supplier#000008480
Supplier#000007398	V8eE6oZ00OFNU,	4sSDA4ACReklNjEm5T6b	Supplier#000008532
Supplier#000007402	4UVv58erylrjmqSR5	Uc29q4,5xVdDOF87UZrxhr4xWS0ihEUXuh	Supplier#000008595
Supplier#000007448	yhhpWcatiJi7EJ6Q5VCAQ	MH0iB73GQ3z UW3O DbCbqmc	Supplier#000008610
Supplier#000007477	9m9j0wfhWzCvVHxkU,PpAxwSH0h	SgVgP90vP452sUNTgzL9zKwXHXAzV6tV	Supplier#000008705
Supplier#000007509	q8,V6LJR0HJjHcOusG7aLTMg	aE,trRNdPx,4yinTD903DebDIp	Supplier#000008742
Supplier#000007561	rMcFg2530VC	HmPlQEzKCPEctUL14,kKq	Supplier#000008841
Supplier#000007789	rQ7cUcPrtudOyO3svNSkimqH6qrfWT2Sz	I 85Lulsekbg2xrSIzm0	Supplier#000008895
Supplier#000007801	69fi,U1r6enUb	2cH4okfaLSZTTg8sKRbbJQxkmeFu2Esj	Supplier#000008967
Supplier#000007818	yhhc2CQec Jrvc8zqBi83	2kwEHyMG 7FwozNImAUE6mH0hYtqYculJM	Supplier#000008972
Supplier#000007885	u3sicchh5ZpyTUPn1cJKNcAoabIWgY	w2vF6 D5YZO3visPXsqVfLADTK	Supplier#000009032
Supplier#000007918	r,v9mBQ6LoEYyj1	qK,trB6Sdy4Dz1BRUFNy	

```

Supplier#000009147      |rOAuryHxpZ9eOvx      |SELECT
Supplier#000009252      |F7cZaPUHwh1 ZKyj3xmAVWC1XdP uelp5m,i |S_NAME,
Supplier#000009278      |RqYTzgxj93CLX 0mcYfCENOfD |COUNT(*) AS NUMWAIT
Supplier#000009327      |uoqMdf7e7Gj9dbQ53    |FROM
Supplier#000009430      |igRqmneFt            |SUPPLIER,
Supplier#000009567      |r4Wfx4c3xsEAjcgJ71HHZByorn1 D9vrztXlv4|LINEITEM L1,
Supplier#000009601      |51m637bO,Rw5DnHWFUvLacRx9 |ORDERTBL,
Supplier#000009709      |rRnCbHYgDgl9PZYnyWKVYSUW0vKg |NATION
Supplier#000009753      |wLhVEcRmd7PkJF4FBnGK7Z |WHERE
Supplier#000009796      |z,y4Idmr15DOvPUqYG   |S_SUPPKEY = L1.L_SUPPKEY
Supplier#000009799      |4wNjXGa4OKWl        |AND O_ORDERKEY = L1.L_ORDERKEY
Supplier#000009811      |E3iuyq7UnZxU7oPZIE2Gu6|AND O_ORDERSTATUS='F'
Supplier#000009812      |APFRMy3lCbgFga53n5t9DxzFPQPgnjrGt32 |AND L1.L_RECEIPTDATE > L1.L_COMMITDATE
Supplier#000009862      |rJzweWeN58          |AND EXISTS (
                                SELECT
                                *
                                FROM
                                LINEITEM L2
                                WHERE
                                L2.L_ORDERKEY = L1.L_ORDERKEY
                                AND L2.L_SUPPKEY <> L1.L_SUPPKEY
                                )
                                AND NOT EXISTS (
                                SELECT
                                *
                                FROM
                                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;

[ROWS CUT]

Submitting SQL Request #21:

/* Teradata
*/

/* Teradata
*/

/* @(#)TERADATA 21.sql 1.1.1.1@(#) */
/* Qurey 21 - Var_0 Rev_01 - TPC-H/TPC-R The Suppliers Who Kept
Orders Waiting
Query */

/* Return the first 100 selected rows.
*/

Query 21 complete, 411 rows returned

only displaying 100 rows per TPC-H spec

|Coll_____|_____Col2|
|Supplier#000002829|20|

```

Supplier#000005808		18	Supplier#000008695		14
Supplier#00000262		17	Supplier#000009224		14
Supplier#00000496		17	Supplier#00000357		13
Supplier#000002160		17	Supplier#00000436		13
Supplier#000002301		17	Supplier#00000610		13
Supplier#000002540		17	Supplier#00000788		13
Supplier#000003063		17	Supplier#00000889		13
Supplier#000005178		17	Supplier#000001062		13
Supplier#000008331		17	Supplier#000001498		13
Supplier#000002005		16	Supplier#000002056		13
Supplier#000002095		16	Supplier#000002312		13
Supplier#000005799		16	Supplier#000002344		13
Supplier#000005842		16	Supplier#000002596		13
Supplier#000006450		16	Supplier#000002615		13
Supplier#000006939		16	Supplier#000002978		13
Supplier#000009200		16	Supplier#000003048		13
Supplier#000009727		16	Supplier#000003234		13
Supplier#000000486		15	Supplier#000003727		13
Supplier#000000565		15	Supplier#000003806		13
Supplier#000001046		15	Supplier#000004472		13
Supplier#000001047		15	Supplier#000005236		13
Supplier#000001161		15	Supplier#000005906		13
Supplier#000001336		15	Supplier#000006241		13
Supplier#000001435		15	Supplier#000006326		13
Supplier#000003075		15	Supplier#000006384		13
Supplier#000003335		15	Supplier#000006394		13
Supplier#000005649		15	Supplier#000006624		13
Supplier#000006027		15	Supplier#000006629		13
Supplier#000006795		15	Supplier#000006682		13
Supplier#000006800		15	Supplier#000006737		13
Supplier#000006824		15	Supplier#000006825		13
Supplier#000007131		15	Supplier#000007021		13
Supplier#000007382		15	Supplier#000007417		13
Supplier#000008913		15	Supplier#000007497		13
Supplier#000009787		15	Supplier#000007602		13
Supplier#000000633		14	Supplier#000008134		13
Supplier#000001960		14	Supplier#000008234		13
Supplier#000002323		14	Supplier#000009435		13
Supplier#000002490		14	Supplier#000009436		13
Supplier#000002993		14	Supplier#000009564		13
Supplier#000003101		14	Supplier#000009896		13
Supplier#000004489		14	Supplier#000000379		12
Supplier#000005435		14	Supplier#000000673		12
Supplier#000005583		14	Supplier#000000762		12
Supplier#000005774		14	Supplier#000000811		12
Supplier#000007579		14	Supplier#000000821		12
Supplier#000008180		14	Supplier#000001337		12

```

|Supplier#000001916      |          12|
|Supplier#000001925      |          12|
|Supplier#000002039      |          12|
|Supplier#000002357      |          12|
|Supplier#000002483      |          12|

```

```

) AS CUSTSALE
GROUP BY
  CENTRYCODE
ORDER BY
  CENTRYCODE;

```

Submitting SQL Request #22:

Query 22 complete, 7 rows returned

```

/* Teradata
*/
/* @(#)TERADATA 22.sql 1.1.1.1@(#) */
/* Query 22 - Var_0 Rev_01 - TPC-H/TPC-R Global Saleds
opportunity Query */

```

Col1	Col2	Col3
13	888	6737713.99
17	861	6460573.72
18	964	7236687.40
23	892	6701457.95
29	948	7158866.63
30	909	6808436.13
31	922	6806670.18

```

SELECT
  CENTRYCODE,
  COUNT(*) AS NUMCUST,
  SUM(C_ACCTBAL) AS TOTACCTBAL
FROM
  (
    SELECT
      SUBSTRING (C_PHONE FROM 1 FOR 2) AS CENTRYCODE,
      C_ACCTBAL
    FROM
      CUSTOMER
    WHERE
      SUBSTRING (C_PHONE FROM 1 FOR 2) IN
        ('13','31','23','29','30','18','17')
      AND C_ACCTBAL > (
        SELECT
          AVG(C_ACCTBAL)
        FROM
          CUSTOMER
        WHERE
          C_ACCTBAL > 0.00
          AND SUBSTRING (C_PHONE FROM 1 FOR
2) IN
('13','31','23','29','30','18','17')
        )
      AND NOT EXISTS (
        SELECT
          *
        FROM
          ORDERTBL
        WHERE
          O_CUSTKEY=C_CUSTKEY
      )
  )

```

Appendix D. Query Substitution Parameters and Seeds Used

```

STREAM00      SEED=1212091008
1      73
2      32      STEEL      MIDDLE EAST
3      HOUSEHOLD      1995-03-26
4      1997-01-01
5      MIDDLE EAST      1994-01-01
6      1994-01-01      0.09      24
7      UNITED STATES      GERMANY
8      GERMANY EUROPE      MEDIUM PLATED NICKEL
9      indian
10     1994-06-01
11     UNITED STATES      0.0000000100
12     SHIP      RAIL      1993-01-01
13     pending packages
14     1993-08-01
15     1994-06-01

```

```

16     Brand#23      LARGE POLISHED      48      31      42
36     20
12     26      27
17     Brand#42      SM BAG
18     314
19     Brand#14      Brand#14      Brand#42      2
11     21
20     navajo      1993-01-01      CHINA
21     EGYPT
22     32      29      16      34      15      19      21

```

```

STREAM01      SEED=1212091009
1      81
2      20      BRASS      AMERICA
3      AUTOMOBILE      1995-03-12
4      1994-10-01
5      AFRICA      1994-01-01
6      1994-01-01      0.07      25
7      MOZAMBIQUE      UNITED STATES
8      UNITED STATES      AMERICA MEDIUM ANODIZED NICKEL
9      ghost
10     1993-03-01
11     JAPAN      0.0000000100
12     FOB      RAIL      1993-01-01
13     pending packages
14     1993-11-01
15     1996-12-01
16     Brand#21      PROMO BRUSHED      30      12      43
13     17
48     18      2
17     Brand#52      SM PACK
18     315
19     Brand#11      Brand#52      Brand#41      7
12     28
20     aquamarine      1997-01-01      GERMANY
21     VIETNAM
22     13      27      28      25      31      18      15

```

```

STREAM02      SEED=1212091010
1      89
2      8      TIN      MIDDLE EAST
3      HOUSEHOLD      1995-03-28
4      1997-05-01
5      AMERICA      1994-01-01
6      1994-01-01      0.04      25
7      INDIA      MOROCCO
8      MOROCCO AFRICA      SMALL POLISHED BRASS

```

9 drab
 10 1993-12-01
 11 ALGERIA 0.0000000100
 12 MAIL RAIL 1994-01-01
 13 pending packages
 14 1994-03-01
 15 1994-09-01
 16 Brand#25 MEDIUM BURNISHED 15 19 6
 24
 31 47 11 12
 17 Brand#22 SM DRUM
 18 313
 19 Brand#13 Brand#35 Brand#45 3
 13 25
 20 lace 1995-01-01 RUSSIA
 21 KENYA
 22 32 17 33 15 20 21 22

STREAM03 SEED=1212091011

1 97
 2 45 COPPER ASIA
 3 AUTOMOBILE 1995-03-14
 4 1995-02-01
 5 ASIA 1995-01-01
 6 1995-01-01 0.02 24
 7 ALGERIA GERMANY
 8 GERMANY EUROPE SMALL BURNISHED BRASS
 9 cream
 10 1994-09-01
 11 JORDAN 0.0000000100
 12 TRUCK RAIL 1994-01-01
 13 pending requests
 14 1994-06-01
 15 1997-04-01
 16 Brand#23 ECONOMY PLATED 40 18 19
 24 28
 39 23 20
 17 Brand#42 LG BAG
 18 315
 19 Brand#25 Brand#23 Brand#34 8
 14 21
 20 slate 1994-01-01 JAPAN
 21 FRANCE
 22 16 17 29 15 20 28 22

STREAM04 SEED=1212091012

1 105

2 33 STEEL MIDDLE EAST
 3 FURNITURE 1995-03-30
 4 1997-09-01
 5 EUROPE 1995-01-01
 6 1995-01-01 0.07 25
 7 PERU UNITED STATES
 8 UNITED STATES AMERICA STANDARD BRUSHED BRASS
 9 chartreuse
 10 1993-07-01
 11 ARGENTINA 0.0000000100
 12 RAIL TRUCK 1994-01-01
 13 pending requests
 14 1994-09-01
 15 1994-12-01
 16 Brand#21 STANDARD BRUSHED 2 45 8
 9
 27 31 14 25
 17 Brand#12 LG PACK
 18 312
 19 Brand#22 Brand#51 Brand#34 3
 15 28
 20 drab 1997-01-01 BRAZIL
 21 UNITED KINGDOM
 22 12 22 25 13 11 17 19

STREAM05 SEED=1212091013

1 113
 2 21 BRASS ASIA
 3 AUTOMOBILE 1995-03-16
 4 1995-06-01
 5 MIDDLE EAST 1995-01-01
 6 1995-01-01 0.04 25
 7 INDIA MOZAMBIQUE
 8 MOZAMBIQUE AFRICA STANDARD PLATED BRASS
 9 blanched
 10 1994-04-01
 11 KENYA 0.0000000100
 12 REG AIR TRUCK 1994-01-01
 13 pending requests
 14 1994-12-01
 15 1997-07-01
 16 Brand#25 LARGE ANODIZED 19 45 31
 33 8
 41 6 29
 17 Brand#32 LG DRUM
 18 314

19 Brand#24 Brand#44 Brand#33 9
 16 24
 20 peru 1996-01-01 MOZAMBIQUE
 21 MOROCCO
 22 19 20 28 21 16 13 32

STREAM06 SEED=1212091014

1 60
 2 9 NICKEL AFRICA
 3 FURNITURE 1995-03-01
 4 1993-03-01
 5 AFRICA 1995-01-01
 6 1995-01-01 0.02 24
 7 ALGERIA INDIA
 8 INDIA ASIA STANDARD ANODIZED BRASS
 9 antique
 10 1995-01-01
 11 BRAZIL 0.0000000100
 12 SHIP TRUCK 1995-01-01
 13 unusual requests
 14 1995-03-01
 15 1995-04-01
 16 Brand#23 PROMO PLATED 25 5 15
 12 8
 37 1 44
 17 Brand#52 MED BAG
 18 315
 19 Brand#31 Brand#22 Brand#22 4
 17 20
 20 brown 1994-01-01 FRANCE
 21 GERMANY
 22 16 18 15 12 27 20 24

STREAM07 SEED=1212091015

1 68
 2 46 COPPER ASIA
 3 MACHINERY 1995-03-18
 4 1995-10-01
 5 AMERICA 1996-01-01
 6 1996-01-01 0.07 24
 7 PERU ALGERIA
 8 ALGERIA AFRICA PROMO POLISHED STEEL
 9 turquoise
 10 1993-10-01
 11 MOROCCO 0.0000000100
 12 FOB TRUCK 1995-01-01
 13 unusual requests

14 1995-07-01
 15 1997-11-01
 16 Brand#21 SMALL POLISHED 25 26 49
 42 13
 8 3 39
 17 Brand#22 MED PACK
 18 313
 19 Brand#33 Brand#55 Brand#21 9
 18 28
 20 maroon 1997-01-01 VIETNAM
 21 UNITED STATES
 22 31 17 14 10 29 16 27

STREAM08 SEED=1212091016

1 76
 2 34 STEEL AFRICA
 3 BUILDING 1995-03-03
 4 1993-07-01
 5 EUROPE 1996-01-01
 6 1996-01-01 0.05 25
 7 INDONESIA PERU
 8 PERU AMERICA PROMO BURNISHED STEEL
 9 snow
 10 1994-07-01
 11 CANADA 0.0000000100
 12 MAIL TRUCK 1995-01-01
 13 unusual accounts
 14 1995-10-01
 15 1995-07-01
 16 Brand#25 ECONOMY ANODIZED 29 4
 34 13
 42 9 19 43
 17 Brand#42 MED DRUM
 18 314
 19 Brand#35 Brand#43 Brand#21 4
 19 24
 20 turquoise 1996-01-01 IRAQ
 21 PERU
 22 17 15 16 19 31 34 21

STREAM09 SEED=1212091017

1 84
 2 22 BRASS EUROPE
 3 MACHINERY 1995-03-20
 4 1996-01-01
 5 MIDDLE EAST 1996-01-01
 6 1996-01-01 0.02 24

```

7      ARGENTINA      INDONESIA
8      INDONESIA      ASIA      ECONOMY BRUSHED STEEL
9      sandy
10     1993-05-01
11     MOZAMBIQUE      0.0000000100
12     TRUCK      MAIL      1995-01-01
13     unusual accounts
14     1996-01-01
15     1993-04-01
16     Brand#23      STANDARD BURNISHED      13      3      8
17
18
19     10      35      17
20     Brand#12      JUMBO BAG
21     312
22     Brand#42      Brand#21      Brand#15      10
23     20
24     green      1994-01-01      ALGERIA
25     INDONESIA
26     20      19      17      30      12      18      22

```

PFAST utility uses standard Teradata call-level interface (CLIV2) to communicate the data to the SUT.

The second method involves using the standard version of DBGEN in conjunction with it's in line load option (-d). We replaced the stub in line load routines that came with DBGEN with a routine that automatically invokes the standard Teradata load utility FASTLOAD, and uses a named pipe to transfer the data as generated by DBGEN to FASTLOAD. FASTLOAD then uses the standard Teradata call-level interface (CLIV2) to communicate the data to the SUT.

While either method can be used to load any or all of the TPC-D database, the second method is easy to use and easy to understand, while the first method is somewhat faster. Because of this, the first method was used for all tables, with the exception of Nation and Region, while the second method was used for the remaining tables.

E.1 Inmod for FastLoad used by Pfast

The Inmod subroutine is made up of a number of pieces. The file blkexit.c is the main body of the subroutine, and is a modification of the driver.c code from the standard DBGEN. Tdinload.c is a set of utility routines used by blkexit.c. The files build.c, bm_utils.c, rnd.c, print.c, bcd2.c, and speed_seed.c are used as-is from the standard DBGEN distribution with no modifications. All header files from DBGEN are used as-is from the standard DBGEN distribution with no modifications.

Makefile

```

# Scsid:      @(#)makefile 2.1.2.2
# makefile for Teradata DSS benchamrk data generator
#
#####
# Changes
# -----
# 990106      (jms) Integration of V2 changes
# 981210      (jms) Unification of inline and pfast makefiles
#
#           Still need to include dependencies/sources for
slave.o
#
#####
#BEFORE YOU START
#####
# to compile with varchar uncomment the following line

```

Appendix E. Data Load Components

Description of the loading method

For this TPC-D benchmark, two methods of loading data were employed. The first method uses a freely available parallel load utility called PFAST combined with a version of the DBGEN data generator redesigned to be a subroutine called by PFAST for each row to be loaded. Such subroutines are referred to as "Inmod" routines. The


```

#VARCHAR= -DVC
# to compile for TPCR (N_NAME vs. N_NATIONKEY) uncomment the
following line
NATION= -DTPCR
# to compile with inserts order then lineitem uncomment the
following line
#INSORDERFIRST= -IOL
# to compile with deletes order then lineitem uncomment the
following line
#DEORDERFIRST= -DOL
#####
## CHANGE NAME OF ANSI COMPILER HERE
#####
CC      = cc
LDLFLAGS = -g
LIBS    = -lm -lc
# Current values for DATABASE are: INFORMIX, DB2, TDAT (Teradata)
DATABASE= TDAT
PLATFORM= AT
#
LOCALIZE= -D$(DATABASE) -D$(PLATFORM)
CFLAGS = -g $(LOCALIZE) $(VARCHAR) $(NATION) $(INSORDERFIRST)
$(DEORDERFIRST) -o1 -586 -I/usr/include -I/usr/ucbinclude
#
# NO CHANGES SHOULD BE NECESSARY BELOW THIS LINE
#####
#
PROG1 = dbgen
PROG2 = qgen
PROG3 = tpccdriver
PROG4 = customer supplier part partsupp order lineitem nation
region
PROGS = $(PROG1) $(PROG2) $(PROG3) $(PROG4)
#
HDR1 = dss.h rnd.h config.h dsstypes.h shared.h bcd2.h
SRC1 = build.c driver.c bm_utils.c rnd.c print.c tdatload.c
bcd2.c \
    speed_seed.c permute.c text.c common.c
OBJ1 = build.o driver.o bm_utils.o rnd.o print.o bcd2.o \
    speed_seed.o tdatload.o permute.o text.o common.o
#
SRC2 = qgen.c varsub.c
HDR2 = tpcd.h permute.h
OBJ2 = build.o bm_utils.o qgen.o rnd.o varsub.o permute.o text.o
speed_seed.o
#
HDR3 = tdatsql.h tsq1t.h sql.h sq1ext.h sq1types.h

```

```

SRC3 = tpccdriver.c tdatsql.c tsq1t.c
OBJ3 = build.o tpccdriver.o bm_utils.o rnd.o vsub.o bcd2.o
tdatsql.o tsq1t.o \
    tdatload.o permute.o text.o common.o speed_seed.o
LIB3 = -lcliv2 -ltdusr -lsocket -lnsl -lx
#
HDR4 = pf.h
SRC4 = blkexit.c tdinload.c fastslav.c tester.c
OBJ4 = build.o bm_utils.o rnd.o print.o tdinload.o bcd2.o
speed_seed.o \
    fastslav.o text.o permute.o common.o
LIB4 = -lcliv2 -ltdusr -lsocket -lnsl -lgen
#
HDRS = $(HDR1) $(HDR2) $(HDR3) $(HDR4)
SRCS = $(SRC1) $(SRC2) $(SRC3) $(SRC4)
OBS = $(OBJ1) $(OBJ2) $(OBJ3) $(OBJ4)
JUNK = part.o partsupp.o order.o lineitem.o supplier.o customer.o
nation.o \
    region.o tester
#
SETS = dists.dss
DDL = dss.ddl dss.ri
OTHER=makefile $(SETS) $(DDL)
DOC=README HISTORY PORTING.NOTES
#
DBGNSRC=$(SRCS) $(HDRS) $(OTHER) $(DOC)
ALLSRC=$(SRCS) $(HDRS) $(OTHER) $(DOC)
#
##
# PROGRAM TARGETS
##
all: $(PROGS)
pfast: $(PROG4)
tester: order.o tester.c $(OBJ4)
    $(CC) $(CFLAGS) -DPFAST -o tester tester.c lineitem.o \
        build.o bm_utils.o rnd.o print.o tdinload.o bcd2.o
\
    speed_seed.o text.o common.o permute.o \
    -lcliv2 -ltdusr -lsocket -lnsl -lm -lc -lgen
#
$(PROG1): $(OBJ1) $(SETS)
    $(CC) $(LDLFLAGS) -o $@ $(OBJ1) $(LIBS)
$(PROG2): $(OBJ2) permute.h
    $(CC) $(LDLFLAGS) -o $@ $(OBJ2) $(LIBS)

```

```

$(PROG3): $(OBJ3) $(SETS) permute.h
    $(CC) $(LDFLAGS) -o $@ $(OBJ3) $(LIBS) $(LIB3)

$(PROG4): $$@.o $(OBJ4)
    $(CC) $(LDFLAGS) -o $@ $$@.o $(OBJ4) $(LIB4) $(LIBS)
##
# Utility Targets
##
clean:
    rm -f $(PROGS) $(OBJS) $(JUNK)

lint:
    lint $(CFLAGS) $(LDFLAGS) $(SRC3)

tar: $(DBGENSRC)
    tar cvhf $(PROG1).tar $(DBGENSRC)

dbgenshar: $(DBGENSRC)
    shar -o dbgen.shar $(DBGENSRC)

dbgendisk: $(DBGENSRC)
    for f in $(DBGENSRC) ; \
    do \
        unix2dos $$f /pcfs/$$f ; \
    done

portable:
    for f in $(DBGENSRC) ; \
    do \
        expand $$f > /tmp/foo; \
        mv -f /tmp/foo $$f; \
        awk 'length > 72 { print FILENAME ":" NR " too long " }'
    done

release: $(ALLSRC) .release
    for f in $(ALLSRC) ; \
    do \
        get -e -r`cat .branch` -b SCCS/s.$$f ; \
        delta -r`cat .release`.1 -y"creating a new release"
    done
SCCS/s.$$f ; \
done
mv .release .branch

##
# Dependencies
##
rnd.o: rnd.h
vsub.o: varsub.c
    $(CC) $(CFLAGS) -c -o vsub.o -DLOAD_LOADER varsub.c
$(OBJ1): config.h makefile
$(OBJ1): dss.h dsstypes.h
$(OBJ2): dss.h tpcd.h config.h
$(OBJ3): dss.h tpcd.h config.h dsstypes.h tdatsql.h

```

```

$(OBJ4): dss.h rnd.h config.h dsstypes.h shared.h bcd2.h
# inmod dependencies
fastslav.o: fastslav.c
    $(CC) -c -O fastslav.c
lineitem.o: blkexit.c dss.h dsstypes.h config.h blkexit.c
    $(CC) $(CFLAGS) -DPFAST -DTABLE=LINE -o $@ -c blkexit.c
order.o: blkexit.c dss.h dsstypes.h config.h blkexit.c
    $(CC) $(CFLAGS) -DPFAST -DTABLE=ORDER -o $@ -c blkexit.c
part.o: blkexit.c dss.h dsstypes.h config.h blkexit.c
    $(CC) $(CFLAGS) -DPFAST -DTABLE=PART -o $@ -c blkexit.c
partsupp.o: blkexit.c dss.h dsstypes.h config.h blkexit.c
    $(CC) $(CFLAGS) -DPFAST -DTABLE=PSUPP -o $@ -c blkexit.c
supplier.o: blkexit.c dss.h dsstypes.h config.h blkexit.c
    $(CC) $(CFLAGS) -O -DPFAST -DTABLE=SUPP -o $@ -c blkexit.c
customer.o: blkexit.c dss.h dsstypes.h config.h blkexit.c
    $(CC) $(CFLAGS) -O -DPFAST -DTABLE=CUST -o $@ -c blkexit.c
region.o: blkexit.c dss.h dsstypes.h config.h blkexit.c
    $(CC) $(CFLAGS) -O -DPFAST -DTABLE=REGION -o $@ -c
blkexit.c
nation.o: blkexit.c dss.h dsstypes.h config.h blkexit.c
    $(CC) $(CFLAGS) -O -DPFAST -DTABLE=NATION -o $@ -c
blkexit.c
$(ALLSRC):
    get -s -r`cat .branch` SCCS/s.$@

```

blkexit.c

```

/* @(#)blkexit.c      2.1.2.2 */
/*****
This is a Teradata-specific version of the DBGEN program for TPC-
D.
This file replaces driver.c from the original DBGEN program.
This
file is used in conjunction with tdatload.c, which replaces the
file
load_stub.c.  load_stub.c is supposed to be customized for each
DB.

```

Note that there are NO teradata specific changes made to any of the header (.h) or other source (.c) files, including those that actually generate the data (such as build.c).

Replacing driver.c is necessary because rather than a main program, we want this routine to be a subroutine that can be called by the

Teradata load utility (FASTLOAD). Such routines are called "INMOD"s.

Although we are replacing driver.c, much of the driver.c source is embedded in this file unchanged. I've marked what parts are copied straight from driver.c, and which parts are Teradata specific changes.

Since an FASTLOAD can only load one table per run, this INMOD needs to know which file to generate rows for. This is handled by a define called "TABLE". It must be defined to be one of the table constants PART,PSUPP,SUPP,CUST,ORDER,LINE,TIME,NATION,REGION. It can be defined as a compiler option -DTABLE=CUST, or it defaults to LINE via the define just below this comment.

Also, because we have a special version of FASTLOAD for loading large amounts of data (called PFAST) that requires slightly different INMODs, you can define PFAST before compiling to get an inmod designed for PFAST. Note that PFAST has a strange restriction: INMODs must NOT write to stdout. So I changed all printf's to go to stderr.

Because we aren't a main program, we can't expect command-line arguments, so we read the information we need from a file called "inmod.info" in the current directory. This file needs to contain three numbers: The scale factor, the total number of child processes being run, and which step we are generating for. These are the same as the command line arguments -s, -C, and -S. If the step is 0 or -1, we generate the entire table.

*****/

```
#ifndef TABLE
#define TABLE LINE
#endif
```

```
/** The following code is from driver.c with no Teradata
modifications: */
/* main driver for dss benchmark */
```

```
#define DECLARER /* EXTERN references get
defined here */
#define NO_FUNC (int (*) ()) NULL /* to clean up tdefs */
#define NO_LFUNC (long (*) ()) NULL /* to clean up
tdefs */

#include "config.h"
#include <stdlib.h>
#if (defined(_POSIX_)||!defined(WIN32)) /* Change for
Windows NT */
#include <unistd.h>
#include <sys/wait.h>
#endif /* WIN32 */
#include <stdio.h> /* */
#include <limits.h>
#include <math.h>
#include <ctype.h>
#include <signal.h>
#include <string.h>
#include <time.h>
#ifdef HP
#include <strings.h>
#endif
#if (defined(_POSIX_)||!defined(WIN32)) /* Change for Windows
NT */
#include <unistd.h>
#include <sys/wait.h>
#include <sys/times.h>
#ifndef CLK_TCK
#define CLK_TCK _sysconf(3) /* 3B2 clock ticks per second */
/* 3 is _SC_CLK_TCK */
#endif
#endif
#if (defined(WIN32)&&!defined(_POSIX_))
#include <process.h>
#include <signal.h>
#include <errno.h>

#endif

/* Ok, here are a few lines added for Teradata */
/* Some extra header files */
#include <stdlib.h>
#include <time.h>

#ifdef WIN32
/* Stuff needed to get process-specific times */
```

```

typedef unsigned long DWORD;
typedef int BOOL;
#define WINAPI __stdcall
#define VOID void
#define DECLSPEC_IMPORT __declspec(dllimport)
#define STRICT
typedef __int64 LONGLONG;
typedef unsigned __int64 DWORDLONG;

typedef void *HANDLE;

#define WINBASEAPI DECLSPEC_IMPORT

typedef struct _FILETIME {
    DWORD dwLowDateTime;
    DWORD dwHighDateTime;
} FILETIME, *PFILETIME, *LPFILETIME;

WINBASEAPI
HANDLE
WINAPI
GetCurrentProcess(
    VOID
);

WINBASEAPI
BOOL
WINAPI
GetProcessTimes(
    HANDLE hProcess,
    LPFILETIME lpCreationTime,
    LPFILETIME lpExitTime,
    LPFILETIME lpKernelTime,
    LPFILETIME lpUserTime
);

#endif
char date[40];

#undef VERSION
#include <coptypes.h> /* a Teradata specific include file */
/* undef VERSION which coptypes.h defined, since config.h will
define it*/
#undef VERSION

#define EM_OK 0
#define FILEERR 400
#define FILEEOF 401

```

```

#define ROWSIZE 32750
#define MAXRECLEN 260

#ifdef WIN32
#pragma pack(1)
#endif

Int32 result;
typedef struct inmod_struct {
#ifdef PFAST
    Int16 Length;
#else
    Int32 ReturnCode;
    Int32 Length;
#endif
    char Body[ROWSIZE];
} inmdtyp,*inmdptr;
inmdptr inmodptr;

#ifdef WIN32
#pragma pack(4)
#endif

#ifdef PFAST
char DummyBuf[32767];

//FILE *jobptr;
#define EOBUFF -1
#endif
char checkpointfile[255];

/* p1 is the pointer to the place in the parcel we are building
where the next column should be copied. */
char *p1;

long i = 1;
long reccnt = 0; /* Record generated by inmod */
long lineitem_in_order = 99999; /* State tracking */
long supplier_in_part = 99999;
long nextrec = 0; /* next record to generate */

size_t templen;

long chkpnt;

```

```
FILE * f2; /* For reading and writing checkpoint information
*/
```

```
#ifndef WIN32
struct tms starttime, endtime; /* For timings */
time_t timeStart, timeEnd, usertime, systime;
#else
FILETIME ProcCreated, ProcExited, StartKernelTime,
StartUserTime, EndKernelTime, EndUserTime;
time_t timeStart, timeEnd;
double usertime, systime;
#endif

/* End of Teradata specific changes, back to standard DBGEN
driver.c code */

#include "dss.h"
#include "dsstypes.h"
#include "bcd2.h"

/*
* Function prototypes
*/
void usage (void);
int prep_direct (char *);
int close_direct (void);
void kill_load (void);
int pload (int tbl);
void gen_tbl (int tnum, long start, long count, long upd_num);
int pr_drange (int tbl, long min, long cnt, long num);
int set_files (int t, int pload);
int partial (int, int);
void gen_seeds (int start, int s);

extern int optind, opterr;
extern char *optarg;

long rowcnt = 0,
     extra_rows = 0,
     minrow = 0;

long upd_num = 0;
double flt_scale;
#if (defined(WIN32)&&!defined(_POSIX_))
char *spawn_args[25];
#endif
#endif
```

```
/*
* general table descriptions. See dss.h for details on structure
* NOTE: tables with no scaling info are scaled according to
* another table
*
*
* the following is based on the tdef structure defined in dss.h
as:
* typedef struct
* {
* char *name; -- name of the table;
* flat file output in <name>.tbl
* long base; -- base scale rowcount of table;
* 0 if derived
* int (*header) (); -- function to prep output
* int (*loader[2]) (); -- functions to present output
* long (*gen_seed) (); -- functions to seed the RNG
* int child; -- non-zero if there is an
associated
detail table
* } tdef;
*
*/

/*
* flat file print functions; used with -F(lat) option
*/
int pr_cust (customer_t * c, int mode);
int pr_line (order_t * o, int mode);
int pr_order (order_t * o, int mode);
int pr_part (part_t * p, int mode);
int pr_psupp (part_t * p, int mode);
int pr_supp (supplier_t * s, int mode);
int pr_order_line (order_t * o, int mode);
int pr_part_psupp (part_t * p, int mode);
int pr_nation (code_t * c, int mode);
int pr_region (code_t * c, int mode);

/*
* inline load functions; used with -D(irect) option
*/
int ld_cust (customer_t * c, int mode);
int ld_line (order_t * o, int mode);
int ld_order (order_t * o, int mode);
```

```

int ld_part (part_t * p, int mode);
int ld_psupp (part_t * p, int mode);
int ld_supp (supplier_t * s, int mode);
int ld_order_line (order_t * o, int mode);
int ld_part_psupp (part_t * p, int mode);
int ld_nation (code_t * c, int mode);
int ld_region (code_t * c, int mode);

/*
* seed generation functions; used with '-O s' option
*/
long sd_cust (long skip_count);
long sd_line (long skip_count);
long sd_order (long skip_count);
long sd_part (long skip_count);
long sd_psupp (long skip_count);
long sd_supp (long skip_count);

/*
* header output functions; used with -h(eader) option
*/
int hd_cust (FILE * f);
int hd_line (FILE * f);
int hd_order (FILE * f);
int hd_part (FILE * f);
int hd_psupp (FILE * f);
int hd_supp (FILE * f);
int hd_order_line (FILE * f);
int hd_part_psupp (FILE * f);
int hd_nation (FILE * f);
int hd_region (FILE * f);

/*
* data verification functions; used with -O v option
*/
int vrf_cust (customer_t * c, int mode);
int vrf_line (order_t * o, int mode);
int vrf_order (order_t * o, int mode);
int vrf_part (part_t * p, int mode);
int vrf_psupp (part_t * p, int mode);
int vrf_supp (supplier_t * s, int mode);
int vrf_order_line (order_t * o, int mode);
int vrf_part_psupp (part_t * p, int mode);
int vrf_nation (code_t * c, int mode);
int vrf_region (code_t * c, int mode);

tdef tdefs[] =
{
{"part.tbl", "part table", 200000, hd_part,
 {pr_part, ld_part}, sd_part, vrf_part,
PSUPP, 0},
{"partsupp.tbl", "partsupplier table", 200000,
hd_psupp,
 {pr_psupp, ld_psupp}, sd_psupp, vrf_psupp,
NONE, 0},
{"supplier.tbl", "suppliers table", 10000, hd_supp,
 {pr_supp, ld_supp}, sd_supp, vrf_supp, NONE,
0},
{"customer.tbl", "customers table", 150000,
hd_cust,
 {pr_cust, ld_cust}, sd_cust, vrf_cust, NONE,
0},
{"orders.tbl", "order table", 150000, hd_order,
 {pr_order, ld_order}, sd_order, vrf_order,
LINE, 0},
{"lineitem.tbl", "lineitem table", 150000, hd_line,
 {pr_line, ld_line}, sd_line, vrf_line, NONE,
0},
{"orders.tbl", "order/lineitem tables", 150000,
hd_order_line,
 {pr_order_line, ld_order_line}, sd_order,
vrf_order_line, LINE, 0},
{"part.tbl", "part/partsupplier tables", 200000,
hd_part_psupp,
 {pr_part_psupp, ld_part_psupp}, sd_part,
vrf_part_psupp, NONE, 0},
{"nation.tbl", "nation table", NATIONS_MAX,
hd_nation,
 {pr_nation, ld_nation}, NO_LFUNC,
vrf_nation, NONE, 0},
{"region.tbl", "region table", NATIONS_MAX,
hd_region,
 {pr_region, ld_region}, NO_LFUNC,
vrf_region, NONE, 0}
};

int *pids;
void mk_sparse(long i, DSS_HUGE *res, long seq);

/*
* re-set default output file names
*/
int

```

```

set_files(int i, int pload)
{
    char line[80], *new_name;

    if (table & (1 << i))
child_table:
    {
        if (pload != -1)
            sprintf(line, "%s.%d", tdefs[i].name, pload + 1);
        else
        {
            fprintf(stderr, "Enter new destination for %s data: ",
                tdefs[i].name);
            if (fgets(line, sizeof(line), stdin) == NULL)
                return(-1);
            if ((new_name = strchr(line, '\n')) != NULL)
                *new_name = '\0';
            if (strlen(line) == 0)
                return(0);
        }
        new_name = (char *)malloc(strlen(line) + 1);
        MALLOC_CHECK(new_name);
        strcpy(new_name, line);
        tdefs[i].name = new_name;
        if (tdefs[i].child != NONE)
        {
            i = tdefs[i].child;
            tdefs[i].child = NONE;
            goto child_table;
        }
    }

    return(0);
}

/*
 * read the distributions needed in the benchamrk
 */
void
load_dists (void)
{
    read_dist (env_config (DIST_TAG, DIST_DFLT), "p_cntr",
&p_cntr_set);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "colors",
&colors);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "p_types",
&p_types_set);

    read_dist (env_config (DIST_TAG, DIST_DFLT), "nations",
&nations);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "regions",
&regions);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "o_oprio",
&o_priority_set);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "instruct",
&l_instruct_set);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "smode",
&l_smode_set);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "category",
&l_category_set);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "rflag",
&l_rflag_set);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "msegmnt",
&c_mseg_set);

    /* load the distributions that contain text generation */
    read_dist (env_config (DIST_TAG, DIST_DFLT), "nouns",
&nouns);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "verbs",
&verbs);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "adjectives",
&adjectives);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "adverbs",
&adverbs);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "auxillaries",
&auxillaries);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "terminators",
&terminators);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "articles",
&articles);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "prepositions",
&prepositions);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "grammar",
&grammar);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "np", &np);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "vp", &vp);
}

/*
 * generate a particular table
 */

```

```

/* This has been changed so that instead of generating the whole
table,
  this routine generates one row for the table and then returns.
  nextrec is used to track which record we are generating.
  order and part must be static, since each contain multiple
rows */

```

```

order_t      o;
part_t       part;
supplier_t   supp;
customer_t   cust;
DSS_HUGE *sk;

```

```

void
gen_h_tbl (int tnum, long sf, long c, long step, long upd_num)
{

```

```

  static DSS_HUGE *i;
  static DSS_HUGE *count;
  static long rem;
  static int init = 0;

```

```

  if (init == 0)
  {

```

```

    INIT_HUGE(i);
    INIT_HUGE(count);
    INIT_HUGE(sk);
    INIT_HUGE(o.okey);
    for (rem=0; rem < O_LCNT_MAX; rem++)
      INIT_HUGE(o.l[rem].okey);
    init = 1;

```

```

    LONG2HUGE(tdefs[tnum].base, i);

```

```

    HUGE_MUL(i, sf);

```

```

    rem = (long)HUGE_MOD(i, c); /* rem holds the extra rows
for the last child */

```

```

    HUGE_DIV(i, c);          /* i holds the step size */

```

```

    HUGE_SET(i, count);

```

```

    HUGE_MUL(i, step);

```

```

    if (step == c - 1)      /* the last child picks up the
remainder */

```

```

      HUGE_ADD(count, rem, count);

```

```

      HUGE_ADD(i, 1, i);    /* i holds the start point */

```

```

#ifdef LASTHUNK /* tweek to do only last 500 */

```

```

  HUGE_SUB (count, 70312000, count);

```

```

  HUGE_ADD (i, 70312000, i);

```

```

#endif

```

```

}

```

```

if (tnum == LINE)

```

```

  {
    lineitem_in_order++;
    if (o.lines>lineitem_in_order)

```

```

      {

```

```

        /* We already have it generated */

```

```

        ld_line (&o, upd_num);

```

```

        return;

```

```

      }

```

```

    else

```

```

      lineitem_in_order=0;

```

```

    }

```

```

if (HUGE_CMP (count, 0) > 0)

```

```

  {

```

```

    nextrec++;

```

```

    if (((nextrec % 10000) == 0) && verbose) fprintf(stderr,

```

```

    ".");

```

```

    HUGE_SET (i, sk);

```

```

    HUGE_DIV (sk, 1 << SPARSE_KEEP);

```

```

    HUGE_MUL (sk, 1 << SPARSE_BITS);

```

```

    HUGE_ADD (sk, upd_num, sk);

```

```

    HUGE_MUL (sk, 1 << SPARSE_KEEP);

```

```

#ifdef SUPPORT_64BITS

```

```

  bcd2_bin(&rem, *i);

```

```

  rem &= (1 << SPARSE_KEEP) - 1;

```

```

  HUGE_ADD (sk, rem, sk);

```

```

  bcd2_bin(&rem, *sk);

```

```

  *sk = rem;

```

```

  bcd2_bin(&rem, *(sk+1));

```

```

  *(sk+1) = rem;

```

```

#else

```

```

  rem = (long)(*i & ((1 << SPARSE_KEEP) - 1));

```

```

  HUGE_ADD (sk, rem, sk);

```

```

#endif

```

```

  mk_order (sk, &o);

```

```

  tdefs[tnum].loader[direct] (&o, upd_num);

```

```

  HUGE_SUB (count, 1, count);

```

```

  HUGE_ADD (i, 1, i);

```

```

  }

```

```

  return;
}

```



```

void
gen_tbl(int tnum, long start, long count, long upd_num)
{
    code_t code;
    static int initdone = 0;
    static int completed = 0;
    long i;

    if (initdone==0)
    {
        INIT_HUGE(o.okey);
        INIT_HUGE(sk);

        for (i=0; i < O_LCNT_MAX; i++)
            INIT_HUGE(o.l[i].okey);
        initdone = 1;
    }

#ifdef TESTER
    fprintf(stderr, "into gen_tbl -- 1\n");
#endif
    if (tnum == LINE)
    {
        lineitem_in_order++;
        if (o.lines>lineitem_in_order)
        {
            /* We already have it generated */
            ld_line (&o, upd_num);

            return;
        }
        else
            lineitem_in_order=0;
    }
    else if (tnum == PSUPP)
    {
        supplier_in_part++;
        if (SUPP_PER_PART>supplier_in_part)
        {
            /* We already have it generated */
            ld_psupp (&part, upd_num);

            return;
        }
        else
            supplier_in_part=0;
    }

#ifdef TESTER
    fprintf(stderr, "into gen_tbl -- 2\n");
#endif
    nextrec++;
    i = nextrec + start - 1;
    if (nextrec <= count)
    {
        LIFENOISE(10000, i);
        switch(tnum)
        {
            case LINE:
            case ORDER:
            case ORDER_LINE:
                mk_sparse(i,
                    sk, (upd_num == 0)?0:1 + upd_num/(10000 /
refresh));
#ifdef TESTER
                fprintf(stderr, "into gen_tbl -- 3\n");
#endif
                mk_order(sk, &o);
#ifdef TESTER
                fprintf(stderr, "into gen_tbl -- 4\n");
#endif
                tdefs[tnum].loader[direct] (&o, upd_num);
                break;
            case SUPP:
                mk_supp(i, &supp);
                tdefs[tnum].loader[direct] (&supp, upd_num);
                break;
            case CUST:
                mk_cust(i, &cust);
                tdefs[tnum].loader[direct] (&cust, upd_num);
                break;
            case PSUPP:
            case PART:
            case PART_PSUPP:
                mk_part(i, &part);
                tdefs[tnum].loader[direct] (&part, upd_num);
                break;
            case NATION:
                mk_nation(i, &code);
                tdefs[tnum].loader[direct] (&code, 0);
                break;
        }
    }
}

```

```

        case REGION:
            mk_region(i, &code);
            tdefs[tnum].loader[direct] (&code, 0);
            break;
        default:
            fprintf(stderr,"%s cannot be built
this way yet\n", tdefs[tnum].name);
            exit(1);
            break;
    }
}

/*completed |= 1 << tnum; */
}

void
usage(void)
{
    /* This isn't a main(), so no usage stuff */
}

/*
 * pload() -- handle the parallel loading of tables
 */
#ifdef DOS
int
partial (int tbl, int s)
{
    char fname[80];

    if (verbose)
    {
        fprintf (stderr, "Starting to load stage %d of %d of
%s...",
            s + 1, children, tdefs[tbl].comment);
    }
    if (direct == 0)
        set_files (tbl, s);

    rowcnt = set_state(TABLE, scale, children, s+1, &extra_rows);
    minrow = rowcnt * s + 1;
    /* JMS */ fprintf(stderr, "stage s is loading %d rows starting at
%d\n",
        rowcnt, minrow);
    /*

```

```

rowcnt = tdefs[tbl].base * scale;
extrarows = rowcnt % children;
rowcnt /= children;
minrow = rowcnt * s + 1;
if (s == children - 1)
    rowcnt += extrarows;
    */

if (verbose)
    fprintf (stderr, "done.\n");

return (0);
}

#ifdef UNDEF
int
partial(int tbl, int s)
{
    char fname[80];

    rowcnt = tdefs[tbl].base * scale;
    if (rowcnt % children)
    {
        fprintf(stderr, "'-C' cannot split load equally\n");
        exit(-1);
    }
    else
        rowcnt /= children;
    /*if (verbose) */
    {
        fprintf(stderr, "Starting to load stage %d of %d of
%s...",
            s + 1, children, tdefs[tbl].comment);
    }
    minrow = rowcnt * s + 1;
    if (direct == 0)
        set_files(tbl, s);

    return(0);
}
#endif

#ifdef NEVER

```

```

int
pload(int tbl)
{
    rowcnt = tdefs[tbl].base * scale;
    if (rowcnt % children)
        {
            fprintf(stderr, "'-C' cannot split load equally\n");
            exit(-1);
        }
    else
        rowcnt /= children;
    /* not used by the INMOD */
    return(0);
}
#endif /* !DOS */
#endif

int stream;
/* Because we aren't a main() program, we cant use getopt or
options
on the command line... So, we replace process_options with the
ability to get some of this information from files on disk */
int
process_options(void)
{
    FILE * fptr;
    char   sclfile[] = "inmod.info";

    if ( !(fptr=fopen(sclfile, "r")) ) {
        fprintf(stderr, "open failed -- %s\n", sclfile);
        fprintf(stderr, "You MUST create this file, and specify the
scale factor\n");
        exit(1);
        //return(FILEERR);
    }
    children = 1;
    step = -1;
    fscanf(fptr, "%lf %d %d\n", &flt_scale, &children, &step);
    fclose(fptr);

    if (flt_scale < 1.0)
        {
            int i;

            scale = 1;
            for (i=PART; i <= LINE; i++)
                {

```

```

                tdefs[i].base = (long)(tdefs[i].base *
flt_scale);
                if (tdefs[i].base < 1)
                    tdefs[i].base = 1;
                }
            else
                scale = (long)flt_scale;
            if (scale < 1)
                {
                    fprintf(stderr,
"WARNING: Scale set to its lower bound (1)\n");
                    scale = 1;
                }

            if (children > 999 && step < 0) /* limitation of
current seed file names */
                {
                    fprintf (stderr, "Child process counts of > 999
not supported.\n");
                    exit (1);
                }
            if (children > 1 && step >= 0)
                pids = malloc (children * sizeof (pid_t));

            fprintf(stderr, "\n      SCALE = %f\n", flt_scale);

            if (children <= 1)
                children = 1;

            fprintf(stderr, "      children (total streams) = %d\n",
children);
            if (children == 1)
                step = -1;

            fprintf(stderr, "      Step = %d\n", step);
            if (step >= 0)
                step--;

            return(0);
        }

/* The next routine is an extract of some, but not all, of the
code from DBGEN's driver.c main() routine.  Because we are
going

```

```

to be a subroutine, we aren't going to have a main(), but we
still need to run some of this initialization code */

int
main_stuff()
{
    int i;

    /* table = (1 << CUST) |
       (1 << SUPP) |
       (1 << NATION) |
       (1 << REGION) |
       (1 << PART_PSUPP) |
       (1 << ORDER_LINE); */

    /* This tells us what table to generate. Inmods can only do
one at a time.*/
    table = (1 << TABLE);

    force = 0;
    verbose = 1;
    columnar = 0;
    header = 0;
    direct = 1;
    scale = 1;
    flt_scale = 1.0;
    updates = 0;
    refresh = UPD_PCT;
    resume = -1;
    step = -1;

    /* Because we might need to reinitialize, let's set these here
*/
    tdefs[PART].base = 200000;
    tdefs[PSUPP].base = 200000;
    tdefs[SUPP].base = 10000;
    tdefs[CUST].base = 150000;
    tdefs[ORDER].base = 150000;
    tdefs[LINE].base = 150000;
    tdefs[ORDER_LINE].base = 150000;
    tdefs[PART_PSUPP].base = 200000;

    tdefs[ORDER].base *=
        ORDERS_PER_CUST; /* have to do this after init */
    tdefs[LINE].base *=
        ORDERS_PER_CUST; /* have to do this after init */

```

```

tdefs[ORDER_LINE].base *=
    ORDERS_PER_CUST; /* have to do this after init */
fnames = 0;
db_name = NULL;
gen_sql = 0;
gen_rng = 0;
children = 1;
minrow = 1;

load_dists();
i= process_options();
if (i!=0)
{
    fprintf(stderr, "Warning, initialization failed to work
right %d\n", i);
    /*return(i);*/
}

fprintf(stderr,
    "TPC %s Population Generator (Version %d.%d.%d%c)\n",
        NAME, VERSION,RELEASE, MODIFICATION,PATCH);
fprintf(stderr, "Copyright %s %s\n", TPC, C_DATES);

/* have to do this after init */
tdefs[NATION].base = nations.count;
tdefs[REGION].base = regions.count;
o.lines = 0;

/**
** actual data generation section starts here
**/
/*
* open database connection or set all the file names, as
appropriate
*/
if (direct)
    prep_direct((db_name)?db_name:DBNAME);
else
    if (fnames)
        for (i=PART; i <= REGION; i++)
            {
                if (table & (1 << i))
                    if (set_files(i, -1))
                        {
                            fprintf(stderr, "Load aborted!\n");
                            exit(1);
                        }
            }

```

```

    }
}
/* So, now we are initialized, and ready to being the
actual loading... At this point, we exit, since we
are a subroutine which will get called once per row
that needs to get generated */

return (0);
}

/* Ok, that's it for source code taken from DBGEN's driver.c.
After this point in the file, everything is Teradata specific
code that is basic INMOD structure. We don't do anything
involving row generation after this point. */

/*****
*
* MakeRecord - Generate a record
*
* Once a row has been successfully built and saved to a row
* buffer, the row buffer is then copied to the main buffer
* and returned to Fastload.
*
*****/
int
MakeRecord()
{
    pl = inmodptr->Body;
    reccnt++; /* This counts which row we are generating */

    if (TABLE >= ORDER && TABLE <= ORDER_LINE && scale >
MAX_32B_SCALE)
        gen_h_tbl(TABLE, scale, children, step, upd_num);
    else
        gen_tbl(TABLE, minrow, rowcnt, upd_num);

    /* end */

#ifdef PFAST
    inmodptr->Length = (Int16)(pl-(char *)inmodptr);
#else

```

```

    inmodptr->Length = pl-inmodptr->Body;
#endif

    if ((recnt % 100000 == 0 )&&(inmodptr->Length>0)) {
        fprintf(stderr, " *** INMOD sent %ld to fastload\n",
recnt);
    }

#ifdef PFAST
    inmodptr->ReturnCode = 0;
    if (inmodptr->Length==0)
#else
    if (inmodptr->Length<=2)
#endif
    {
        if (inmodptr->Length<0 || inmodptr->Length > 32760)
            fprintf(stderr, " !!! Bug!, inmodptr->Length =
%d\n",inmodptr->Length);

#ifdef PFAST
        if (inmodptr->Length >= MAXRECLEN)
            fprintf(stderr, " !!! Bug!, inmodptr->Length =
%d\n",inmodptr->Length);
#endif
        inmodptr->Length = 0;
        recnt--;
        fprintf(stderr, " *** INMOD reached End of file\n");
        fprintf(stderr, " *** INMOD sent %d records to
fastload\n",recnt);
        timeEnd = time(NULL);
        strftime(date, 35, "%a %b %d %H:%M:%S %Z %Y",
localtime(&timeEnd));
        fprintf(stderr, " Finish making records: %s\n", date);

#ifdef WIN32
        if (times(&endtime) == -1) {
            fprintf(stderr, "times() fails\n");
        }
        else
            if
(!GetProcessTimes(GetCurrentProcess(),&ProcCreated,&ProcExited,&E
ndKernelTime,&EndUserTime)) {
                fprintf(stderr, "GetProcessTimes() fails\n");
            }
#endif

        //exit(1);
    } else {

```

```

#ifdef WIN32
    usertime = endtime.tms_utime - starttime.tms_utime;
    systime = endtime.tms_stime - starttime.tms_stime;
    fprintf(stderr,"user time = %10.2lf secs, system time
= %10.2lf secs\n",
        ((double) usertime / CLK_TCK), ((double)
systime / CLK_TCK));
    fprintf(stderr,"time per request = %10.10lf secs
\n",
        ( ( (double) (usertime + systime) /
reccnt) / CLK_TCK ));
#else
    /* Times are in 100 nanosecond units! */
    usertime = (*(LONGLONG*)&EndUserTime -
*(LONGLONG*)&StartUserTime)/10000000.0;
    systime = (*(LONGLONG*)&EndKernelTime -
*(LONGLONG*)&StartKernelTime)/10000000.0;
    fprintf(stderr,"user time = %10.2f secs, system time =
%10.2lf secs\n",
        ( usertime ), ( systime ));
    fprintf(stderr,"time per request = %10.6lf
microsecs \n",
        ( ( (usertime + systime)*1000000.0 /
reccnt) ));
#endif

    }
    fprintf(stderr,"Clock time per request = %10.6lf millisecs
\n",difftime(timeEnd,timeStart)*1000.0/ reccnt);

#ifdef PFAST
    inmodptr->ReturnCode = FILEEOF;
#endif
    return(FILEEOF);
}

return(EM_OK);

Int32 OpenSource()
{
    int rc;

#ifdef PFAST
    if(getenv("HOSTNUMB") == NULL) {
        fprintf(stderr, "can not file HOSTNUMB var");
        return(FILEEOF);
    }
    else
    {
        rc = main_stuff(); /* Initialize everything */
        if (rc!=0)
            fprintf(stderr,"Main routine initialization failure\n");

        if (children > 1 && TABLE <= LINE)
        {
            fprintf(stderr," Generating partial data for partitioned
load\n");
            if (step >= 0)
                partial(TABLE, step);
            else
            {
                fprintf(stderr," Don't know which step I'm supposed
to load\n");
                exit(1);
            }
        }
        else
        {
            if (TABLE < LINE)
                return(FILEEOF);
        }
    }
    fprintf(checkpointfile, "./checkpoint.%ld",
atoi(getenv("HOSTNUMB")) );
    f2 = fopen(checkpointfile,"wb");
#else
    f2 = fopen("./checkpoint.dat","wb");
#endif

    if (f2==0)
    {
        fprintf(stderr,"Can't open the checkpoint file\n");
#ifdef PFAST
        inmodptr->ReturnCode = FILEEOF;
#endif
        return(FILEEOF);
    }

    reccnt = 0L;
    lineitem_in_order = 99999; /* State tracking */
    supplier_in_part = 99999;
    nextrec = 0;

    fwrite(&reccnt,4,1,f2); /* mark that we are at the start */
    fclose(f2);
    f2 = NULL;

    rc = main_stuff(); /* Initialize everything */
    if (rc!=0)
        fprintf(stderr,"Main routine initialization failure\n");

    if (children > 1 && TABLE <= LINE)
    {
        fprintf(stderr," Generating partial data for partitioned
load\n");
        if (step >= 0)
            partial(TABLE, step);
        else
        {
            fprintf(stderr," Don't know which step I'm supposed
to load\n");
            exit(1);
        }
    }
    else
    {
        if (TABLE < LINE)
            return(FILEEOF);
    }
}

```

```

        rowcnt = tdefs[TABLE].base * scale;
    else
        rowcnt = tdefs[TABLE].base;
}

if (verbose)
    fprintf(stderr, "Generating data for %s [pid: %d] \n",
            tdefs[TABLE].comment,

DSS_PROC );

        timeEnd = timeStart = time(NULL);
        strftime(date, 35, "%a %b %d %H:%M:%S %Z %Y",
localtime(&timeEnd));
#ifdef WIN32
        if (times(&starttime) == -1) {
            fprintf(stderr, "times() fails\n");
            // exit(1);
        }
#else
        if
(!GetProcessTimes(GetCurrentProcess(), &ProcCreated, &ProcExited, &S
tartKernelTime, &StartUserTime)) {
            fprintf(stderr, "GetProcessTimes() fails\n");
            //exit(1);
        }
#endif

        fprintf(stderr, "\n Start making records: %s\n", date);

        return(0);
}

/*****
 * HostRestart - Host restarted, rest and start sending data
 *               from the beginning.
*****/
Int32 HostRestart()
{
    int oldchkpnt;

#ifdef PFAST

```

```

        /* Notice how I use DummyBuf! For restart calls the user
doesn't */
        /* pass a big buffer for me to fill, I gotta use one on the
stack. */
        /* If I don't then GetReco() is gonna die like a stuck pig!
*/

        inmodptr = (struct inmod_struct *) DummyBuf;

        if(getenv("HOSTNUMB") == NULL) {
            fprintf(stderr, "can not file HOSTNUMB var");
            return(FILEOF);
        }
        sprintf(checkpointfile, "./checkpoint.%ld",
atoi(getenv("HOSTNUMB"))) );
        f2 = fopen(checkpointfile, "rb");
    #else
        f2 = fopen("./checkpoint.dat", "rb");
    #endif

        if (f2==0)
        {
            fprintf(stderr, "Can't open the checkpoint
file\n");
#ifdef PFAST
                inmodptr->ReturnCode = FILEOF;
            #endif
                return(FILEOF);
        }
        fread(&oldchkpnt, 4, 1, f2); /* mark that we are at
the start */
        fclose(f2);
        f2 = NULL;

        chkpnt = oldchkpnt;

        result = OpenSource();
        if (result!=EM_OK)
            return(result);

        recnt = 0;
        lineitem_in_order = 99999; /* State tracking */
        supplier_in_part = 99999;
        nextrec = 0;

```

```

while(reccnt < oldchkpnt)
{
    result = MakeRecord();
    if (result!=EM_OK)
        return(result);
}

return(EM_OK);
}

/*****
 * CheckPoint - Save checkpoint
 *****/
Int32 CheckPoint()
{
    chkpnt = reccnt;

#ifdef PFAST
    f2 = fopen(checkpointfile,"wb");
#else
    f2 = fopen("./checkpoint.dat","wb");
#endif

    if (f2==0)
    {
        fprintf(stderr,"Can't open the checkpoint
file\n");
#ifdef PFAST
        inmodptr->ReturnCode = FILEOF;
#endif
        return(FILEOF);
    }
    fwrite(&chkpnt,4,1,f2); /* mark that we are at
the start */
    fclose(f2);
    f2 = NULL;
    return(EM_OK);
}

/*****
 * DBCRestart - DBC restarted, do what you have to do.
 *****/

```

```

Int32 DBCRestart()
{
    int oldchkpnt;

#ifdef PFAST
    /* Notice how I use DummyBuf! For restart calls the user
doesn't */
    /* pass a big buffer for me to fill, I gotta use one on the
stack. */
    /* If I don't then GetReco() is gonna die like a stuck pig!
*/

    inmodptr = (struct inmod_struct *) DummyBuf;
    f2 = fopen(checkpointfile,"rb");
#else
    f2 = fopen("./checkpoint.dat","rb");
#endif

    if (f2==0)
    {
        fprintf(stderr,"Can't open the checkpoint
file\n");
#ifdef PFAST
        inmodptr->ReturnCode = FILEOF;
#endif
        return(FILEOF);
    }
    fread(&oldchkpnt,4,1,f2); /* mark that we are at
the start */
    fclose(f2);
    f2 = NULL;

    if (oldchkpnt!=chkpnt)
    {
        fprintf(stderr," **** Something is wrong trying to
restart at checkpoint\n");
    }

    result = OpenSource();
    if (result!=EM_OK)
        return(result);
    reccnt = 0;
    while(reccnt < oldchkpnt)
    {

```



```

        result = MakeRecord();
        if (result!=EM_OK)
            return(result);
    }

    return(EM_OK);
}

/*****
 * CleanUp - Do cleanup.
*****/
Int32 CleanUp()
{
    if (verbose)
        fprintf(stderr, "done.\n");

    if (direct)
        close_direct();
    if (f2)
        fclose(f2);
    fprintf(stderr, " *** INMOD Done\n");
    return(EM_OK);
}

/*****
 * InvalidCode - Invalid inmod code returned.
*****/
Int32 InvalidCode()
{
    fprintf(stderr, " *** Invalid Inmod code\n");
    return(EM_OK);
}

/*****
 *
 * BLKEXIT - Start processing
 *
 * This is the main module which contains the checks for
 * number of records generated and buffer filling. This
 * module also sends the filled buffer to the DBC.
 *
*****/

```

```

#ifdef PFAST
Int32 BLKEXIT(int Function, char * tblptr, Int32 BufLen)
#else
Int32 BLKEXIT(char * tblptr)
#endif
{
    Int32 result;

    inmodptr = (struct inmod_struct *)tblptr;

#ifdef PFAST
    switch (Function) {
#else
    switch (inmodptr->ReturnCode) {
#endif
        case 0: result=OpenSource();
                if (EM_OK==result)
                    fprintf(stderr, " *** INMOD starting...\n");
                else
                    break;

        case 1:
#ifdef PFAST
            if (BufLen < MAXRECLEN + 2)
                return(EOBUF); /* not enough room
in buffer */
#endif
            result = MakeRecord();
            break;
        case 2: result = HostRestart();
                break;
        case 3: result = CheckPoint();
                break;
        case 4: result = DBCRestart();
                break;
        case 5: result = CleanUp();
                break;
        default: result = InvalidCode();
    }

    return(result);
}

```

tdinload.c

```
/*
*****
* Title:      load_stub.c
* Sccsid:    @(#)tdinload.c    2.1.2.2
* Description:
*           stub routines for:
*           inline load of dss benchmark
*           header creation for dss benchmark
*
*****
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>

#ifdef WIN32
#define INLINE _inline
#else
#define INLINE _Inline
#include <unistd.h>
#endif
#include "config.h"
#include "dss.h"
#include "dsstypes.h"

/* Start of Teradata specific code... These are the routines from
   load_stub.c, which are intended to be customized. */

/*char buffer[32760]; */
```

```
extern char *p1;
char * p1;

extern long reccnt; /* Record generated by
inmod */
extern long lineitem_in_order; /* State tracking */
extern long supplier_in_part;
extern long nextrec;

/* First, some helper routines to copy data into the parcel we
are building */
INLINE
void DR_Strt()
{
/* p1 = buffer + 2;*/ /* Skip 2 for the length field */
if (p1==NULL)
fprintf(stderr," Error: tdinload.c called when p1
is null");
}
INLINE
void DR_End(int t)
{
if(t);
}

INLINE
void DR_Int(long x)
{
memcpy(p1,&x,4);
p1 += 4;
}

/* Used only for orderkey, which gets > 32 bits at times */
INLINE
static void DR_Huge(DSS_HUGE * data)
{
/*
* below 300G, 32 bits will hold all orderkeys. If the platform
support 64b,
* then trust the compiler's cast to allow storage of just 32b.
If the platform
* doesn't support 64 bit data types, then they are implemented
as a pair of
```

```

* longs, with the LSBs in the first long, so the pointer
reference is still
* ok.
*
* above 300G, 64 bit platforms are still fine. 32 bit platforms
have to work
* harder. I have not tested this code, but am told that the 32
bit support
* worked. (jms, 981124)
*/
    long value;

    if (scale < 300)
        {
            value = (long)*data; /* below 300G 32b is
sufficient */
            memcpy(p1,&value,4);
            p1+=4;
        }
    else
#ifdef SUPPORT_64BITS
        {
            memcpy(p1,data,8);
            p1+=8;
        }
#else
    {
        double tempx;
        unsigned long templ;
        long temph;
        if (*(data+1) ==0)
            {
                memcpy(p1,*data,4);
                p1 += 4;
                memcpy(p1,*(data+1),4);
                p1 += 4;
            }
        else
            {
                /* Sigh, don't have an Int64 type, so need
to use double and convert to 2 longs */
                tempx = *(data+1)*10000000.0+*data;
                temph = (long)(tempx/4294967296.0);
                templ = (unsigned long)(tempx-
tempx*4294967296.0);
                memcpy(p1,&temph,4);
                p1 += 4;
            }
    }

```

```

        memcpy(p1,&tempx,4);
        p1 += 4;
    }
}
#endif
}

INLINE
void DR_VStr(char * x)
{
    short len;
    len=(short)strlen(x);
    memcpy(p1,&len,2);
    p1 += 2;
    memcpy(p1,x,len);
    p1 += len;
}
INLINE
void DR_Str(char * x,size_t y)
{
    memset(p1,' ',y);
    memcpy(p1,x,strlen(x));
    p1+=y;
}
INLINE
void DR_Money(long x)
{
    long temp=0;

    if (x<0)
        temp = -1;
    /*sprintf(p1, "%13.2f", x * 0.01);
p1+=13; */
    memcpy(p1,&x,4);
    p1 += 4;
    memcpy(p1,&temp,4);
    p1 += 4;
}
INLINE
void DR_Char(char x)
{
    *p1 = x; p1++;
}
INLINE
void DR_Date(char * x)

```

```

{
    long tempdate;
    int y,m,d;
    /*memset(pl, ' ',10);
    memcpy(pl,x,strlen(x));
    pl+=10;*/
    y=atoi(x);
    m=atoi(x+5);
    d=atoi(x+8);
/*
    if (y<1800 || y>2200 || m<1 || m>12 || d<1 || d>31)
        {
            fprintf(stderr," Help! I can't interpret the date
%s\n",x);
            exit(1);
        } */
    tempdate=(y-1900)*10000+m*100+d;
    memcpy(pl,&tempdate,4);
    pl += 4;
}

/*now the routines from load_stub.c */

int
close_direct(void)
{
    fprintf(stderr," **** Done with direct loading ****\n");
    return(0);
}

int
prep_direct(char * dbnamestr)
{
    /* any preload prep goes here */
    if(dbnamestr);
    return(0);
}

int
hd_cust (FILE *f)
{
    static int count = 0;
    if (f);

    if (! count++)

```

```

        fprintf(stderr,"No header has been defined for the
customer table\n");

        return(0);
}

/*
CREATE TABLE TPCD.CUSTOMER ( C_CUSTKEY      INTEGER NOT NULL,
                             C_NAME         VARCHAR(25) NOT NULL,
                             C_ADDRESS      VARCHAR(40) NOT NULL,
                             C_NATIONKEY    INTEGER NOT NULL,
                             C_PHONE        CHAR(15) NOT NULL,
                             C_ACCTBAL      DECIMAL(15,2)   NOT
NULL,
                             C_MKTSEGMENT   CHAR(10) NOT NULL,
                             C_COMMENT      VARCHAR(117) NOT
NULL);
*/

int
ld_cust (customer_t *cp, int mode)
{
    if(mode);
    DR_Strt();
    DR_Int(cp->custkey);
    DR_VStr(cp->name);
    DR_VStr(cp->address);
    DR_Int(cp->nation_code);
    DR_Str(cp->phone,PHONE_LEN);
    DR_Money(cp->acctbal);
    DR_Str(cp->mktsegment,10);
    DR_VStr(cp->comment);
    DR_End(CUST);
    return(0);
}

int
hd_part (FILE *f)
{
    static int count = 0;
    if(f);

    if (! count++)
        fprintf(stderr,"No header has been defined for the part
table\n");

```

```

        return(0);
    }
}
/*
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   DECIMAL(15,2) NOT NULL,
                        P_COMMENT       VARCHAR(23) NOT NULL );
*/
int
ld_part (part_t *pp, int mode)
{
    if(mode);
    DR_Strt();
    DR_Int(pp->partkey);
    DR_VStr(pp->name);
    DR_Str(pp->mfg, P_MFG_LEN);
    DR_Str(pp->brand, P_BRND_LEN);
    DR_VStr(pp->type);
    DR_Int(pp->size);
    DR_Str(pp->container, P_CNTR_LEN);
    DR_Money(pp->retailprice);
    DR_VStr(pp->comment);
    DR_End(PART);
    return(0);
}
/*
CREATE TABLE TPCD.PARTSUPP ( PS_PARTKEY      INTEGER NOT NULL,
                              PS_SUPPKEY      INTEGER NOT NULL,
                              PS_AVAILQTY     INTEGER NOT NULL,
                              PS_SUPPLYCOST   DECIMAL(15,2) NOT
NULL,
                              PS_COMMENT      VARCHAR(199) NOT NULL
);
*/
int
ld_psupp (part_t *pp, int mode)
{

```

```

    int i;
    if(mode);
    i = supplier_in_part;
    /*for (i = 0; i < SUPP_PER_PART; i++) */
    {
        DR_Strt();
        DR_Int(pp->s[i].partkey);
        DR_Int(pp->s[i].suppkey);
        DR_Int(pp->s[i].qty);
        DR_Money(pp->s[i].scost);
        DR_VStr(pp->s[i].comment);
        DR_End(PSUPP);
    }
    return(0);
}
int
hd_time (FILE *f)
{
    static int count = 0;
    if(f);

    if (! count++)
        fprintf(stderr, "No header has been defined for the time
table\n");

    return(0);
}
int
hd_supp (FILE *f)
{
    static int count = 0;
    if(f);

    if (! count++)
        fprintf(stderr, "No header has been defined for the
supplier table\n");

    return(0);
}
/*
CREATE TABLE TPCD.SUPPLIER ( S_SUPPKEY      INTEGER NOT NULL,
                              S_NAME         CHAR(25) NOT NULL,

```

```

                S_ADDRESS      VARCHAR(40) NOT NULL,
                S_NATIONKEY     INTEGER NOT NULL,
                S_PHONE         CHAR(15) NOT NULL,
                S_ACCTBAL       DECIMAL(15,2) NOT
NULL,
                S_COMMENT      VARCHAR(101) NOT
NULL);
*/

int
ld_supp (supplier_t *sp, int mode)
{
    if(mode);
    DR_Strt();
    DR_Int(sp->suppkey);
    DR_Str(sp->name,25);
    DR_VStr(sp->address);
    DR_Int(sp->nation_code);
    DR_Str(sp->phone,15);
    DR_Money(sp->acctbal);
    DR_VStr(sp->comment);
    DR_End(SUPP);
    return(0);
}

int
hd_order (FILE *f)
{
    static int count = 0;
    if(f);

    if (! count++)
        fprintf(stderr,"No header has been defined for the order
table\n");

    return(0);
}

/*
CREATE TABLE TPCD.ORDER ( O_ORDERKEY      INTEGER NOT NULL,
                O_CUSTKEY      INTEGER NOT NULL,
                O_ORDERSTATUS  CHAR(1) NOT NULL,
                O_TOTALPRICE   DECIMAL(15,2) 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);
*/

int
ld_order (order_t *p, int mode)
{
    if(mode);
    DR_Strt();

    DR_Huge(p->okey);

    DR_Int(p->custkey);
    DR_Char(p->orderstatus);
    DR_Money(p->totalprice);
    DR_Date(p->odate);
    DR_Str(p->opriority,O_OPRIO_LEN);
    DR_Str(p->clerk,O_CLRK_LEN);
    DR_Int(p->spriority);
    DR_VStr(p->comment);
    DR_End(ORDER);
    return(0);
}

/*
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     DECIMAL(15,2) NOT
NULL,
                L_EXTENDEDPRICE DECIMAL(15,2) NOT
NULL,
                L_DISCOUNT    DECIMAL(15,2) NOT
NULL,
                L_TAX          DECIMAL(15,2) 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,

```

```

- R          L_SHIPINSTRUCT CHAR(25) NOT NULL, -          "No header has been defined for the",
- R          L_SHIPMODE     CHAR(10) NOT NULL, -          "part supplier table");
- R          L_COMMENT      VARCHAR(44) NOT   }
NULL);
*/
ld_line (order_t *p, int mode)
{
    int i;
    if(mode);
    i = lineitem_in_order;
    /*for (i = 0; i < p->lines; i++) */
    {
        DR_Strt();
        DR_Huge(p->l[i].okey);

        DR_Int(p->l[i].partkey);
        DR_Int(p->l[i].suppkey);
        DR_Int(p->l[i].lcnt);
        DR_Int(p->l[i].quantity);
        DR_Money(p->l[i].eprice);
        DR_Money(p->l[i].discount);
        DR_Money(p->l[i].tax);
        DR_Char(p->l[i].rflag[0]);
        DR_Char(p->l[i].lstatus[0]);
        DR_Date(p->l[i].sdate);
        DR_Date(p->l[i].cdate);
        DR_Date(p->l[i].rdate);
        DR_Str(p->l[i].shipinstruct, L_INST_LEN);
        DR_Str(p->l[i].shipmode, L_SMODE_LEN);
        DR_VStr(p->l[i].comment);
        DR_End(LINE);
    }
    return(0);
}

int
hd_line (FILE *f)
{
    static int count = 0;
    if(f);

    if (! count++)
        fprintf(stderr, "No header has been defined for the
lineitem table\n");

    return(0);
}

int
hd_nation (FILE *f)
{
    static int count = 0;
    if(f);

    if (! count++)
        fprintf(stderr, "No header has been defined for the nation
table\n");

    return(0);
}

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

int
ld_line (order_t *p, int mode)
{
    if(mode);
    DR_Strt();
    DR_Int(cp->code);
    DR_Str(cp->text, NATION_LEN);
}

int
ld_nation (code_t *cp, int mode)
{
    if(mode);
    DR_Strt();
    DR_Int(cp->code);
    DR_Str(cp->text, NATION_LEN);
}

```

```

        DR_Int(cp->join);
        DR_VStr(cp->comment);
        DR_End(NATION);
        return(0);
    }

int
hd_region (FILE *f)
{
    static int count = 0;
    if(f);

    if (! count++)
        fprintf(stderr, "No header has been defined for the region
table\n");

    return(0);
}

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

int
ld_region (code_t *cp, int mode)
{
    if(mode);
    DR_Strt();
    DR_Int(cp->code);
    DR_Str(cp->text, REGION_LEN);
    /* DR_Int(cp->join); *** ac4 */
    DR_VStr(cp->comment);
    DR_End(REGION);

    return(0);
}

int
ld_order_line (order_t *p, int mode)
{

    ld_order(p, mode);
    ld_line (p, mode);

    return(0);
}

```

```

    }

int
hd_order_line (FILE *f)
{

    hd_order(f);
    hd_line (f);

    return(0);
}

int
ld_part_psupp (part_t *p, int mode)
{
    ld_part(p, mode);
    ld_psupp (p, mode);

    return(0);
}

int
hd_part_psupp (FILE *f)
{
    hd_part(f);
    hd_psupp(f);

    return(0);
}

```

E.2 Source Code for Pipeline-to-FastLoad Approach

In this approach to loading, DBGEN is basically unmodified from the standard TPC distribution. All that has been done is to replace the load_stub.c file with one specific for Teradata.

tdataload.c


```

/*****
*
* Title:      load_stub.c
* Sccsid:    @(#)tdataload.c    2.1.2.2
* Description:
*           stub routines for:
*           inline load of dss benchmark
*           header creation for dss benchmark
*
*****/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>

#ifdef WIN32
#include <process.h>
#define WIN32_LEAN_AND_MEAN
#define NOATOM
#define NOGDICAPMASKS
#define NOMETAFILE
#define NOMINMAX
#define NOMSG
#define NOOPENFILE
#define NORASTEROPS
#define NOSCROLL
#define NOSOUND
#define NOSYMETRICS
#define NOTEXTMETRIC
#define NOWH
#define NOCOMM
#define NOKANJI
#define NOMCX
#define INLINE _inline
#pragma warning(disable:4201)
#pragma warning(disable:4214)
#pragma warning(disable:4514)
#include <windows.h>
#else
#define INLINE _Inline
#include <unistd.h>
#include <sys/wait.h>
#endif
#include "config.h"

```

```

#include "dss.h"
#include "dsstypes.h"

/* Start of Teradata specific code... These are the routines from
   load_stub.c, which are intended to be customized. */
int prep_direct(char *);

#ifdef WIN32
FILE * pipefile[20];
#else
HANDLE pipehandle[20];
HANDLE processhandle[20];
#endif

char logonstr[256] = { 0 };
char pipename[120];
char fastname[120];
char buffer[32760];
/*extern*/
char * pl;
int reccnt = 0;
int loadinprogress = 0;
int fastloadsstarted = 0;

typedef struct COLUMN_T
{
    int tbl;          /* table */
    char *name;      /* column name */
    char *dtype;    /* data type */
    char *ltype;    /* load type */
    char *format;   /* format qualifier */
    char *other;    /* other qualifier */
} column_t;

/*
 * one entry for each column in the schema. Note: orderkey is
 * purposefully left
 * out of order and lineitem since it changes with scale factor
 */
column_t columns[] =
{
    {CUST,  "C_CUSTKEY",          "INTEGER", "INTEGER", "", ""},
    {CUST,  "C_NAME",
    "VARCHAR(25)", "VARCHAR(25)", "", "CASESPECIFIC"},
    {CUST,  "C_ADDRESS",
    "VARCHAR(40)", "VARCHAR(40)", "", "CASESPECIFIC"},

```

```

{CUST, "C_NATIONKEY", "INTEGER", "INTEGER", "", ""},
#ifdef VC
{CUST, "C_PHONE",
"VARCHAR(15)", "CHAR(15)", "", "CASESPECIFIC"},
#else
{CUST, "C_PHONE",
"CHAR(15)", "CHAR(15)", "", "CASESPECIFIC"},
#endif
{CUST, "C_ACCTBAL", "DECIMAL(15,2)", "DECIMAL(15,2)", "", ""},
#ifdef VC
{CUST, "C_MKTSEGMENT",
"VARCHAR(10)", "CHAR(10)", "", "CASESPECIFIC"},
#else
{CUST, "C_MKTSEGMENT",
"CHAR(10)", "CHAR(10)", "", "CASESPECIFIC"},
#endif
{CUST, "C_COMMENT",
"VARCHAR(117)", "VARCHAR(117)", "", "CASESPECIFIC"},
{LINE, "L_PARTKEY", "INTEGER", "INTEGER", "", ""},
{LINE, "L_SUPPKEY", "INTEGER", "INTEGER", "", ""},
{LINE, "L_LINENUMBER", "INTEGER", "INTEGER", "", ""},
{LINE, "L_QUANTITY", "DECIMAL(15,2)", "INTEGER", "", ""},
{LINE, "L_EXTENDEDPRICE", "DECIMAL(15,2)", "DECIMAL(15,2)", "", ""},
{LINE, "L_DISCOUNT", "DECIMAL(15,2)", "DECIMAL(15,2)", "", ""},
{LINE, "L_TAX", "DECIMAL(15,2)", "DECIMAL(15,2)", "", ""},
#ifdef VC
{LINE, "L_RETURNFLAG",
"VARCHAR(1)", "CHAR(1)", "", "CASESPECIFIC"},
#else
{LINE, "L_RETURNFLAG", "CHAR(1)", "CHAR(1)", "", "CASESPECIFIC"},
#endif
#ifdef VC
{LINE, "L_LINESTATUS",
"VARCHAR(1)", "CHAR(1)", "", "CASESPECIFIC"},
#else
{LINE, "L_LINESTATUS", "CHAR(1)", "CHAR(1)", "", "CASESPECIFIC"},
#endif
{LINE, "L_SHIPDATE", "DATE", "DATE", "FORMAT 'yyyy-mm-dd'", ""},
{LINE, "L_COMMITDATE", "DATE", "DATE", "FORMAT 'yyyy-mm-dd'", ""},
{LINE, "L_RECEIPTDATE", "DATE", "DATE", "FORMAT 'yyyy-mm-dd'", ""},
#ifdef VC
{LINE, "L_SHIPINSTRUCT",
"VARCHAR(25)", "CHAR(25)", "", "CASESPECIFIC"},
#else

```

```

{LINE, "L_SHIPINSTRUCT",
"CHAR(25)", "CHAR(25)", "", "CASESPECIFIC"},
#endif
#ifdef VC
{LINE, "L_SHIPMODE",
"VARCHAR(10)", "CHAR(10)", "", "CASESPECIFIC"},
#else
{LINE, "L_SHIPMODE",
"CHAR(10)", "CHAR(10)", "", "CASESPECIFIC"},
#endif
{LINE, "L_COMMENT",
"VARCHAR(44)", "VARCHAR(44)", "", "CASESPECIFIC"},
{NATION, "N_NATIONKEY", "INTEGER", "INTEGER", "", ""},
#ifdef VC
{NATION, "N_NAME",
"VARCHAR(25)", "CHAR(25)", "", "CASESPECIFIC"},
#else
{NATION, "N_NAME",
"CHAR(25)", "CHAR(25)", "", "CASESPECIFIC"},
#endif
{NATION, "N_REGIONKEY", "INTEGER", "INTEGER", "", ""},
{NATION, "N_COMMENT",
"VARCHAR(152)", "VARCHAR(152)", "", "CASESPECIFIC"},
{ORDER, "O_CUSTKEY", "INTEGER", "INTEGER", "", ""},
#ifdef VC
{ORDER, "O_ORDERSTATUS",
"VARCHAR(1)", "CHAR(1)", "", "CASESPECIFIC"},
#else
{ORDER, "O_ORDERSTATUS", "CHAR(1)", "CHAR(1)", "", "CASESPECIFIC"},
#endif
{ORDER, "O_TOTALPRICE", "DECIMAL(15,2)", "DECIMAL(15,2)", "", ""},
{ORDER, "O_ORDERDATE", "DATE", "DATE", "FORMAT 'yyyy-mm-dd'", ""},
#ifdef VC
{ORDER,
"O_ORDERPRIORITY", "VARCHAR(15)", "CHAR(15)", "", "CASESPECIFIC"},
#else
{ORDER,
"O_ORDERPRIORITY", "CHAR(15)", "CHAR(15)", "", "CASESPECIFIC"},
#endif
#ifdef VC
{ORDER, "O_CLERK",
"VARCHAR(15)", "CHAR(15)", "", "CASESPECIFIC"},
#else
{ORDER, "O_CLERK",
"CHAR(15)", "CHAR(15)", "", "CASESPECIFIC"},
#endif

```

```

{ORDER, "O_SHIPPRIORITY", "INTEGER", "INTEGER", "", ""},
{ORDER, "O_COMMENT",
"VARCHAR(79)", "VARCHAR(79)", "", "CASESPECIFIC"},
{PART, "P_PARTKEY", "INTEGER", "INTEGER", "", ""},
{PART, "P_NAME",
"VARCHAR(55)", "VARCHAR(55)", "", "CASESPECIFIC"},
#ifdef VC
{PART, "P_MFGR",
"VARCHAR(25)", "CHAR(25)", "", "CASESPECIFIC"},
#else
{PART, "P_MFGR",
"CHAR(25)", "CHAR(25)", "", "CASESPECIFIC"},
#endif
#ifdef VC
{PART, "P_BRAND",
"VARCHAR(10)", "CHAR(10)", "", "CASESPECIFIC"},
#else
{PART, "P_BRAND",
"CHAR(10)", "CHAR(10)", "", "CASESPECIFIC"},
#endif
{PART, "P_TYPE",
"VARCHAR(25)", "VARCHAR(25)", "", "CASESPECIFIC"},
{PART, "P_SIZE", "INTEGER", "INTEGER", "", ""},
#ifdef VC
{PART, "P_CONTAINER",
"VARCHAR(10)", "CHAR(10)", "", "CASESPECIFIC"},
#else
{PART, "P_CONTAINER",
"CHAR(10)", "CHAR(10)", "", "CASESPECIFIC"},
#endif
{PART, "P_RETAILPRICE", "DECIMAL(15,2)", "DECIMAL(15,2)", "", ""},
{PART, "P_COMMENT",
"VARCHAR(23)", "VARCHAR(23)", "", "CASESPECIFIC"},
{PSUPP, "PS_PARTKEY", "INTEGER", "INTEGER", "", ""},
{PSUPP, "PS_SUPPKEY", "INTEGER", "INTEGER", "", ""},
{PSUPP, "PS_AVAILQTY", "INTEGER", "INTEGER", "", ""},
{PSUPP, "PS_SUPPLYCOST", "DECIMAL(15,2)", "DECIMAL(15,2)", "", ""},
{PSUPP, "PS_COMMENT",
"VARCHAR(199)", "VARCHAR(199)", "", "CASESPECIFIC"},
{REGION, "R_REGIONKEY", "INTEGER", "INTEGER", "", ""},
#ifdef VC
{REGION, "R_NAME",
"VARCHAR(25)", "CHAR(25)", "", "CASESPECIFIC"},
#else
{REGION, "R_NAME",
"CHAR(25)", "CHAR(25)", "", "CASESPECIFIC"},
#endif

```

```

{REGION, "R_COMMENT",
"VARCHAR(152)", "VARCHAR(152)", "", "CASESPECIFIC"},
{SUPP, "S_SUPPKEY", "INTEGER", "INTEGER", "", ""},
#ifdef VC
{SUPP, "S_NAME",
"VARCHAR(25)", "CHAR(25)", "", "CASESPECIFIC"},
#else
{SUPP, "S_NAME",
"CHAR(25)", "CHAR(25)", "", "CASESPECIFIC"},
#endif
{SUPP, "S_ADDRESS",
"VARCHAR(40)", "VARCHAR(40)", "", "CASESPECIFIC"},
{SUPP, "S_NATIONKEY", "INTEGER", "INTEGER", "", ""},
#ifdef VC
{SUPP, "S_PHONE",
"VARCHAR(15)", "CHAR(15)", "", "CASESPECIFIC"},
#else
{SUPP, "S_PHONE",
"CHAR(15)", "CHAR(15)", "", "CASESPECIFIC"},
#endif
{SUPP, "S_ACCTBAL", "DECIMAL(15,2)", "DECIMAL(15,2)", "", ""},
{SUPP, "S_COMMENT",
"VARCHAR(101)", "VARCHAR(101)", "", "CASESPECIFIC"},
{-1, "", "", ""}
};

int
gen_schema(FILE *fp, int t, int flags)
{
    int i,
        colcount;

    colcount = 0;
    for (i=0; columns[i].tbl >= 0; i++)
        if (columns[i].tbl == t)
            {
                if (t == ORDER || t == LINE)
                    colcount = 1;
                if (colcount)
                    fprintf(fp, ",");
                switch(flags)
                {
                    case 0:
                        fprintf(fp, "%s\n", columns[i].name);
                        break;
                    case 1:
                        fprintf(fp, "F_%s\t(%s)\n",

```

```

                columns[i].name,
                columns[i].ltype);
            break;
        case 2:
            fprintf(fp, ":F_%s\n",
columns[i].name);
            break;
        case 3:
            fprintf(fp, "%s\t%s\t%s NOT NULL
%s\n",
                columns[i].name,
                columns[i].dtype,
                columns[i].format,
                columns[i].other);
            break;
        }
        colcount++;
    }
    return(0);
}

```

```
void WriteFastloadJob(int t,int mode)
{
```

```

    FILE * fastjob;
    char tablename[60];

    tablename[0] = '\0';
    if (t==19)
    {
        strcpy(tablename, "KEYS_TO_DELETE");
    }
    else
    switch(t) {
        case PART: strcpy(tablename, "PARTTBL"); break;
        case PSUPP: strcpy(tablename, "PARTSUPP"); break;
        case SUPP: strcpy(tablename, "SUPPLIER"); break;
        case CUST: strcpy(tablename, "CUSTOMER"); break;
        case ORDER: strcpy(tablename, "ORDERTBL"); break;
        case LINE: strcpy(tablename, "LINEITEM"); break;
        case NATION: strcpy(tablename, "NATION"); break;
        case REGION: strcpy(tablename, "REGION"); break;
        case ORDER_LINE:
        case PART_PSUPP:
        case NONE:
        default: fprintf(stderr, " Illegal table to load\n");
            exit(1);
    }
}

```

```

    }
    if (mode>0 && t!=19)
        strcat(tablename, "_UPDATES");

    if (logonstr[0] == '\0' && db_name[0] != '\0')
        prep_direct(db_name);

    if (verbose)
        fprintf(stderr, " Opening %s for write\n", fastname);
    fastjob = fopen(fastname, "w");
    if (fastjob==NULL)
        {
            fprintf(stderr, " Cannot create the fastload input
job stream for %s\n", fastname);
            return;
        }
    if (fprintf(fastjob, "\n")<=0)
        fprintf(stderr, " Cannot write the fastload input job
stream\n");
    fprintf(fastjob, "\n");
    fprintf(fastjob, "SESSIONS 30;\n");
    fprintf(fastjob, "LOGON %s;\n", logonstr);
    fprintf(fastjob, "\n");
    fprintf(fastjob, "DROP TABLE %s;\n", tablename);
    fprintf(fastjob, "DROP TABLE ERR%s1;\n", tablename);
    fprintf(fastjob, "DROP TABLE ERR%s2;\n", tablename);
    fprintf(fastjob, "\n");
#ifdef WIN32
    fprintf(fastjob, "CREATE TABLE %s\n", tablename);
#else
    fprintf(fastjob, "CREATE MULTISSET TABLE %s\n", tablename);
#endif
    fprintf(fastjob, " (\n");
    switch(t) {
        case 19:
            if (scale>300) /* uncommented by ac4 on 5-31-98 */
                fprintf(fastjob, " ORDERKEY DECIMAL(15,0)
NOT NULL\n");
            else /* this line and following uncommented by ac4 on 5-
31-98 */
                fprintf(fastjob, " ORDERKEY INTEGER NOT
NULL\n");
            fprintf(fastjob, " )\n");
            fprintf(fastjob, "UNIQUE PRIMARY INDEX( ORDERKEY );\n");
            break;
        case CUST:
            gen_schema(fastjob, CUST, 3);
    }
}

```

```

        fprintf(fastjob, "  )\n");
        fprintf(fastjob, "UNIQUE PRIMARY INDEX( C_CUSTKEY );\n");
        break;
case LINE:
    if (scale>300) /* uncommented by ac4 on 5-31-98 */
        fprintf(fastjob, "    L_ORDERKEY          DECIMAL(15,0)
NOT NULL\n");
    else /* this line and the following uncommented by ac4 on
5-31-98 */
        fprintf(fastjob, "    L_ORDERKEY          INTEGER NOT
NULL\n");
        gen_schema(fastjob, LINE, 3);
        fprintf(fastjob, "  )\n");
        fprintf(fastjob, ";\n");
        break;
case NATION:
    gen_schema(fastjob, NATION, 3);
    fprintf(fastjob, "  )\n");
#ifdef TPCR
    fprintf(fastjob, "PRIMARY INDEX( N_NAME )\n");
#else
    fprintf(fastjob, "PRIMARY INDEX( N_NATIONKEY )\n");
#endif
    fprintf(fastjob, ";\n");
    break;
case ORDER:
    if (scale>300) /* uncommented by ac4 on 5-31-98 */
        fprintf(fastjob, "    O_ORDERKEY          DECIMAL(15,0)
NOT NULL\n");
    else /* this line and the following uncommented by ac4 on
5-31-98 */
        fprintf(fastjob, "    O_ORDERKEY          INTEGER NOT
NULL\n");
        gen_schema(fastjob, ORDER, 3);
        fprintf(fastjob, "  )\n");
        fprintf(fastjob, "UNIQUE PRIMARY INDEX( O_ORDERKEY );\n");
        break;
case PART:
    gen_schema(fastjob, PART, 3);
    fprintf(fastjob, "  )\n");
    fprintf(fastjob, "UNIQUE PRIMARY INDEX( P_PARTKEY );\n");
    break;
case PSUPP:
    gen_schema(fastjob, PSUPP, 3);
    fprintf(fastjob, "  )\n");
    /* Default to non-unique primary index on PS_PARTKEY */
    fprintf(fastjob, ";\n");

```

```

        break;
case REGION:
    gen_schema(fastjob, REGION, 3);
    fprintf(fastjob, "  )\n");
    fprintf(fastjob, "UNIQUE PRIMARY INDEX( R_REGIONKEY );\n");
    break;
case SUPP:
    gen_schema(fastjob, SUPP, 3);
    fprintf(fastjob, "  )\n");
    fprintf(fastjob, "UNIQUE PRIMARY INDEX( S_SUPPKEY );\n");
    break;
}

fprintf(fastjob, "\n");
fprintf(fastjob, "BEGIN LOADING %s\n", tablename);
fprintf(fastjob, "    ERRORFILES ERR%s1,
ERR%s2;\n", tablename, tablename);
fprintf(fastjob, "\n");
fprintf(fastjob, "DEFINE\n");
switch (t) {
case 19:
    if (scale>300) /* uncommented by ac4 5-31-98 */
        fprintf(fastjob, "    F_ORDERKEY
(DECIMAL(15,0))\n");
    else /* this line and the following uncommented by ac4 */
        fprintf(fastjob, "    F_ORDERKEY
(INTEGER)\n");
    break;
case LINE:
    if (scale>300) /* uncommented by ac4 on 5-31-98 */
        fprintf(fastjob, "    F_L_ORDERKEY
(DECIMAL(15,0))\n");
    else /* this line and the following uncommented by ac4 on
5-31-98 */
        fprintf(fastjob, "    F_L_ORDERKEY
(INTEGER)\n");
        gen_schema(fastjob, LINE, 1);
        break;
case ORDER:
    if (scale>300) /* this line uncommented by ac4 on 5-31-98
*/
        fprintf(fastjob, "    F_O_ORDERKEY
(DECIMAL(15,0))\n");
    else /* this line and the following uncommented by ac4 on
5-31-98 */
        fprintf(fastjob, "    F_O_ORDERKEY
(INTEGER)\n");

```

```

        gen_schema(fastjob, ORDER, 1);
        break;
case CUST:
case NATION:
case PART:
case PSUPP:
case REGION:
case SUPP:
        gen_schema(fastjob, t, 1);
        break;
default:
        exit(-999); /* bad table name */
        break;
}

fprintf(fastjob, "FILE=%s;\n", pipename);

fprintf(fastjob, "\n");
fprintf(fastjob, "SHOW;\n");
fprintf(fastjob, "\n");
fprintf(fastjob, "INSERT INTO %s\n", tablename);
fprintf(fastjob, " (\n");
switch (t) {
case 19:
        fprintf(fastjob, " ORDERKEY\n");
        fprintf(fastjob, " )\n");
        fprintf(fastjob, "VALUES\n");
        fprintf(fastjob, " (\n");
        fprintf(fastjob, " :F_ORDERKEY\n");
        break;
case LINE:
        fprintf(fastjob, " L_ORDERKEY\n");
        gen_schema(fastjob, LINE, 0);
        fprintf(fastjob, " )\n");
        fprintf(fastjob, "VALUES\n");
        fprintf(fastjob, " (\n");
        fprintf(fastjob, " :F_L_ORDERKEY\n");
        gen_schema(fastjob, LINE, 2);
        break;
case ORDER:
        fprintf(fastjob, " O_ORDERKEY\n");
        gen_schema(fastjob, ORDER, 0);
        fprintf(fastjob, " )\n");
        fprintf(fastjob, "VALUES\n");
        fprintf(fastjob, " (\n");
        fprintf(fastjob, " :F_O_ORDERKEY\n");
        gen_schema(fastjob, ORDER, 2);

```

```

        break;
case NATION:
case CUST:
case PART:
case PSUPP:
case REGION:
case SUPP:
        gen_schema(fastjob, t, 0);
        fprintf(fastjob, " )\n");
        fprintf(fastjob, "VALUES\n");
        fprintf(fastjob, " (\n");
        gen_schema(fastjob, t, 2);
        break;
default:
        exit(-999); /* bad table name */
        break;
}

fprintf(fastjob, " );\n");
fprintf(fastjob, "\n");
fprintf(fastjob, "END LOADING;\n");
fprintf(fastjob, "\n");
fprintf(fastjob, "LOGOFF;\n");

fflush(fastjob);

if (fclose(fastjob)!=0)
        fprintf(stderr, " Error closing fastjob file\n");
if (verbose)
        fprintf(stderr, " Closed %s and ready for fastload to read
it\n", fastname);
}

int
end_load(void)
{
#ifdef WIN32
        char temp[200];
#endif
        if (loadinprogress==0) return(0);

        if (loadinprogress==PART_PSUPP)
        {
#ifdef WIN32
                FlushFileBuffers(pipehandle[PART]);
                //DisconnectNamedPipe(pipehandle[PART]);

```

```

        CloseHandle(pipehandle[PART]);
        FlushFileBuffers(pipehandle[PSUPP]);
        //DisconnectNamedPipe(pipehandle[PSUPP]);
        CloseHandle(pipehandle[PSUPP]);

    WaitForSingleObject(processhandle[PSUPP],9999999);
#else
    fclose(pipefile[PART]);
    fclose(pipefile[PSUPP]);
    system("rm part.tbl.pipe");
    system("rm partsupp.tbl.pipe");
#endif
    fprintf(stderr," Sent all rows for part and
partsupp tables...\n");
    }
    else if (loadinprogress==ORDER_LINE)
    {
#ifdef WIN32
        FlushFileBuffers(pipehandle[ORDER]);
        //DisconnectNamedPipe(pipehandle[ORDER]);
        CloseHandle(pipehandle[ORDER]);
        FlushFileBuffers(pipehandle[LINE]);
        //DisconnectNamedPipe(pipehandle[LINE]);
        CloseHandle(pipehandle[LINE]);

        WaitForSingleObject(processhandle[LINE],9999999);
#else
        fclose(pipefile[ORDER]);
        fclose(pipefile[LINE]);
        system("rm order.tbl.pipe");
        system("rm lineitem.tbl.pipe");
#endif
        fprintf(stderr," Sent all rows for order and
lineitem tables...\n");
    }
    else
    {
#ifdef WIN32
        FlushFileBuffers(pipehandle[loadinprogress]);

        //DisconnectNamedPipe(pipehandle[loadinprogress]);
        CloseHandle(pipehandle[loadinprogress]);

        WaitForSingleObject(processhandle[loadinprogress],9999999)
;
#else
        fclose(pipefile[loadinprogress]);
        strcpy(temp,"rm ");
        strcat(temp,pipename);
        system(temp);
#endif
    }
    if (loadinprogress==19)
    {
        fprintf(stderr," Sent all keys for
deletes ...\n");
        system("rm deletes.fastload");
    }
    else
        fprintf(stderr," Sent all rows for %s
...\n",tdefs[loadinprogress].name);
    }
    if (verbose)
        fprintf(stderr, " Closed the pipe...\n");
    loadinprogress = 0;
    return(0);
}

int
start_load2(int t,int mode)
{
    int rc;

    char temp[120];

#ifdef WIN32
    STARTUPINFO startinfo;
    PROCESS_INFORMATION procinfo;
#else
    pid_t fpid;
#endif

    pipename[0] = '\0';
#ifdef WIN32
    strcpy(pipename,"\\\\.\\pipe\\");
#endif
    if (t==19)
    {
        strcat(pipename,"deletes.pipe");
        strcpy(fastname,"deletes.fastload");
    }
    else

```

```

    {
        strcat(pipename,tdefs[t].name);
        strcat(pipename,".pipe");

        strcpy(fastname,tdefs[t].name);
        strcat(fastname,".fastload");
    }

    WriteFastloadJob(t,mode);

#ifdef WIN32
if (verbose)
    fprintf(stderr," Creating the named pipe... \n");

pipehandle[t] = CreateNamedPipe(pipename,
    PIPE_ACCESS_DUPLEX,
    PIPE_TYPE_BYTE | PIPE_READMODE_BYTE | PIPE_WAIT,
    3,
    32000,
    32000,
    1800000 /* 30 minutes */,NULL);

if (pipehandle[t]==INVALID_HANDLE_VALUE)
    {
        rc = GetLastError();
        fprintf(stderr," CreateNamedPipe error was %d\n",rc);
        exit(rc);
    }

fprintf(stderr," Spawning a copy of fastload...\n");

startinfo.cb = sizeof(STARTUPINFO);
startinfo.lpReserved = NULL;
startinfo.lpDesktop = NULL;
startinfo.lpTitle = NULL;
startinfo.dwFlags = 0;
startinfo.cbReserved2 = 0;
startinfo.lpReserved2 = 0;
startinfo.hStdInput = NULL;
startinfo.hStdOutput = NULL;
startinfo.hStdError = NULL;

//if (! CreateProcess(NULL,"cmd /K fastload < custinmod.txt >
test.out 2>&1",
// NULL,NULL,FALSE,CREATE_NEW_CONSOLE | NORMAL_PRIORITY_CLASS,
    {
        NULL,NULL,&startinfo,&procinfo))
    // {
    //     fprintf(stderr," Spawn of fastload failed,
rc=%d\n",GetLastError());
    //     }

    strcpy(temp,"cmd /K fastload.exe < ");
    strcat(temp,fastname);
    fprintf(stderr,"Running> %s\n",temp);

    if (! CreateProcess(NULL,temp,
        NULL,NULL,FALSE, CREATE_NEW_CONSOLE|NORMAL_PRIORITY_CLASS,
        NULL,NULL,&startinfo,&procinfo))
        {
            rc = GetLastError();
            if (rc==ERROR_FILE_NOT_FOUND)
                fprintf(stderr," Spawn of fastload failed, file not
found\n");
            else
                fprintf(stderr," Spawn of fastload failed,
rc=%d\n",rc);
        }
    processhandle[t] = procinfo.hProcess;

    fprintf(stderr," Waiting for Fastload to connect to the pipe...
\n");
    if (!ConnectNamedPipe(pipehandle[t],NULL))
        {
            rc = GetLastError();
            if (rc==ERROR_CALL_NOT_IMPLEMENTED)
                fprintf(stderr," Can't run this program on Win95
because server side pipes don't exist\n");
            else
                fprintf(stderr," ConnectNamedPipe Error was
%d\n",rc);
            exit(rc);
        }
    fprintf(stderr," Pipe is connected... \n");

#else
    if (verbose)
        fprintf(stderr," Creating the named pipe\n");

    strcpy(temp,"mkfifo ");
    strcat(temp,pipename);

```



```

    if (verbose)
        fprintf(stderr,"%s\n",temp);
    rc= system(temp);
    if (rc!=0)
        {
            if (rc===-1)
                {
                    fprintf(stderr," Error %d
trying to create the pipe\n",errno);
                }
            else
                {
                    if (rc!=256)
                        fprintf(stderr," Create pipe
returned %d\n",rc);
                }
        }

    if (verbose)
        fprintf(stderr," Spawning a copy of fastload for
%s...\n",fastname);
    if((fpid=fork())==0)
        {
            strcpy(temp,"/usr/bin/fastload < ");
            strcat(temp,fastname);
            strcat(temp," > ");
            strcat(temp,fastname);
            strcat(temp,".out 2>&1");
            if (verbose)
                fprintf(stderr,"/usr/bin/sh -c
%s\n",temp);
            execlp("/usr/bin/sh","/usr/bin/sh","-
c",temp,(char *)0);
            fprintf(stderr, "spawn of fastload faild,
%d\n",errno);
            fprintf(stderr, "You must start fastload
manually, using the following command:\n");
            fprintf(stderr, "%s\n",temp);
            exit(errno);
        }
    fastloadsstarted++;
    fprintf(stderr," Spawned a copy of fastload for %s, pid =
%d\n",fastname,fpid);

    if (verbose)
        fprintf(stderr," Connecting to the pipe for %s
(%s)... \n",fastname,pipeiname);

```

```

    pipefile[t] = fopen(pipeiname,"wb");
    if (pipefile[t]==NULL)
        {
            rc = errno;
            fprintf(stderr," Unable to open output file
for direct load\n");
            fprintf(stderr," Error code = %d\n",rc);
            exit(rc);
        }
    else
        {
            if (verbose)
                fprintf(stderr," Opened the pipe for
direct loading of %s\n",fastname);
        }
    }

#endif
    return(0);
}

int
start_load(int t,int mode)
{
    if (loadinprogress!=0) end_load();
    if (t==PART_PSUPP)
        {
            start_load2(PART,mode);
            start_load2(PSUPP,mode);
        }
    else if (t==ORDER_LINE)
        {
            start_load2(ORDER,mode);
            start_load2(LINE,mode);
        }
    else
        start_load2(t,mode);

    loadinprogress = t;
    return(0);
}

/* First, some helper routines to copy data into the parcel we
are building */
static void DR_Strt()

```

```

{
    p1 = buffer + 2;    /* Skip 2 for the length field */
}

static void DR_End(int t)
{
    /* We must write to the pipe with a single I/O call, else
we
could get interleave problems when there are multiple
writers */
#ifdef WIN32
    unsigned long byteswritten;
#endif
    int len;
    int rc;
    reccnt++;

    *p1 = 0x0A; p1++; /* Stick end-of-line in the buffer */
    len = (int)(p1-buffer);

    *((short *)buffer) =(short)(len-3); /* Fill in length */

#ifdef WIN32
    rc = fwrite(buffer,1,len,pipefile[t]);
    if (rc != len)
        {
            fprintf(stderr, " Write to pipe or file
failed, errno=%d.\n",errno);
            exit(1);
        }
    #else
    if
(WriteFile(pipehandle[t],&buffer,len,&byteswritten,NULL))
        {
            if (byteswritten!=(unsigned long)(len))
                {
                    fprintf(stderr, " Byteswritten =
%d\n",byteswritten);
                }
        }
    else
        {
            rc = GetLastError();
            if (rc==ERROR_BROKEN_PIPE)
                fprintf(stderr, " Fastload broke the pipe
prematurely\n");

```

```

else
    fprintf(stderr, " unknown pipe error =
%d\n",rc);
    fprintf(stderr, " when trying to set row
%d\n",reccnt);
    CloseHandle(pipehandle[t]);
    //fclose(pipefile);
    if (rc==0) rc++;
    exit(rc);
}
#endif

INLINE
static void DR_Int(long x)
{
    memcpy(p1,&x,4);
    p1 += 4;
}

/* Used only for orderkey, which gets > 32 bits at times */
INLINE
static void DR_Huge(DSS_HUGE * data)
{
    long value;

/*
* below 300G, 32 bits will hold all orderkeys. If the platform
support 64b,
* then trust the compiler's cast to allow storage of just 32b.
If the platform
* doesn't support 64 bit data types, then they are implemented
as a pair of
* longs, with the LSBs in the first long, so the pointer
reference is still
* ok.
*
* above 300G, 64 bit platforms are still fine. 32 bit platforms
have to work
* harder. I have not tested this code, but am told that the 32
bit support
* worked. (jms, 981124)
*/
    if (scale < 300)
        {

```

```

        value = (long)*data;
        memcpy(p1,&value,4);
        p1+=4;
    }
    else
    {
#ifdef SUPPORT_64BITS
        memcpy(p1,data,8);
        p1+=8;
#else
        double tempx;
        unsigned long templ;
        long temph;
        if (*(data+1) ==0)
        {
            memcpy(p1,*data,4);
            p1 += 4;
            memcpy(p1,(data+1),4);
            p1 += 4;
        }
        else
        {
/* Sigh, don't have an Int64 type; need to use double and convert
to 2 longs */
            tempx = *(data+1)*10000000.0+*data;
            temph = (long)(tempx/4294967296.0);
            templ = (unsigned long)(tempx-
templ*4294967296.0);
            memcpy(p1,&templ,4);
            p1 += 4;
            memcpy(p1,&temph,4);
            p1 += 4;
        }
    }
#endif
    }
}
INLINE
static void DR_VStr(char * x)
{
    short len;
    len=(short)strlen(x);
    memcpy(p1,&len,2);
    p1 += 2;
    memcpy(p1,x,len);
    p1 += len;
}
}

```

```

static void DR_Str(char * x,size_t y)
{
    memset(p1,' ',y);
    memcpy(p1,x,strlen(x));
    p1+=y;
}
INLINE
static void DR_Money(long x)
{
    long temp = 0;
    if (x<0)
        temp = -1;
    /*sprintf(p1, "%13.2f", x * 0.01);
p1+=13; */
    memcpy(p1,&x,4);
    p1 += 4;
    memcpy(p1,&temp,4);
    p1 += 4;
}
INLINE
static void DR_Char(char x)
{
    *p1 = x; p1++;
}
INLINE
static void DR_Date(char * x)
{
    long tempdate;
    int y,m,d;
    /*memset(p1,' ',10);
memcpy(p1,x,strlen(x));
p1+=10;*/
    y=atoi(x);
    m=atoi(x+5);
    d=atoi(x+8);
    if (y<1800 || y>2200 || m<1 || m>12 || d<1 || d>31)
    {
        fprintf(stderr," Help! I can't interpret the date
%s\n",x);
        exit(1);
    }
    tempdate=(y-1900)*10000+m*100+d;
    memcpy(p1,&tempdate,4);
    p1 += 4;
}
}

```

```

/*now the routines from load_stub.c */
int
close_direct(void)
{
#ifdef WIN32
    int c;
    int i;
    int status;
#endif
    if (loadinprogress!=0) end_load();

    loadinprogress = 0;

#ifdef WIN32
    c = fastloadsstarted;
    fprintf(stderr, " Finished sending all rows to all
tables.\n");
    fprintf(stderr, " Now waiting for all fastloads to
end...\n");
    while (c)
    {
        i = wait(&status);
        if (i == -1 && fastloadsstarted)
        {
            fprintf(stderr, "We lost one of the fastloads?\n");
            return(-2);
        }
        if (status & 0xFF)
        {
            if (status & 0xFF == 0117)
                fprintf(stderr, "Fastload Process %d: STOPPED\n",
i);
            else
                fprintf(stderr, "Fastload Process %d: rcvd signal
%d\n",
                    i, status&0x7F);
        }
        c--;
    }
#else
    Sleep(1000);
#endif
#ifdef WIN32
    system("erase *.tbl.pipe");
    system("erase *.tbl.fastload");

```

```

#else
    /*system("rm *.tbl.pipe"); */
    system("rm *.tbl.fastload");
#endif
    fprintf(stderr, " **** Done with direct loading ****\n");
    return(0);
}

int
prep_direct(char * dbnamestr)
{
    fastloadsstarted = 0;
    strcpy(logonstr, dbnamestr);
    if (strcmp(logonstr, "dss")==0 || strcmp(logonstr, "")==0)
    {
        fprintf(stderr, " You MUST use the -n option
to define the Teradata logon string\n");
        exit(1);
    }
    if (verbose)
        fprintf(stderr, " Teradata logon is
%s\n", dbnamestr);
    if (loadinprogress!=0) end_load();

    /* any preload prep goes here */
    return(0);
}

int
hd_cust (FILE *f)
{
    static int count = 0;
    if (f);

    if (! count++)
        printf("No header has been defined for the customer
table\n");

    return(0);
}

/*
CREATE TABLE TPCD.CUSTOMER ( C_CUSTKEY      INTEGER NOT NULL,
                             C_NAME        VARCHAR(25) NOT NULL,
                             C_ADDRESS     VARCHAR(40) NOT NULL,
                             C_NATIONKEY   INTEGER NOT NULL,

```

```

                C_PHONE      CHAR(15) NOT NULL,
                C_ACCTBAL    DECIMAL(15,2)  NOT
NULL,
                C_MKTSEGMENT CHAR(10) NOT NULL,
                C_COMMENT    VARCHAR(117) NOT
NULL);
*/

int
ld_cust (customer_t *cp, int mode)
{
    if(mode);
    if (loadinprogress!=CUST) start_load(CUST,mode);
    DR_Strt();
    DR_Int(cp->custkey);
    DR_VStr(cp->name);
    DR_VStr(cp->address);
    DR_Int(cp->nation_code);
    DR_Str(cp->phone,PHONE_LEN);
    DR_Money(cp->acctbal);
    DR_Str(cp->mktsegment,10);
    DR_VStr(cp->comment);
    DR_End(CUST);
    return(0);
}

int
hd_part (FILE *f)
{
    static int count = 0;
    if(f);

    if (! count++)
        printf("No header has been defined for the part
table\n");

    return(0);
}

/*
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 DECIMAL(15,2) NOT NULL,
                        P_COMMENT      VARCHAR(23) NOT NULL );
*/

int
ld_part (part_t *pp, int mode)
{
    if(mode);
    if (loadinprogress!=PART && loadinprogress!=PART_PSUPP)
start_load(PART,mode);
    DR_Strt();
    DR_Int(pp->partkey);
    DR_VStr(pp->name);
    DR_Str(pp->mfgr,P_MFG_LEN);
    DR_Str(pp->brand,P_BRND_LEN);
    DR_VStr(pp->type);
    DR_Int(pp->size);
    DR_Str(pp->container,P_CNTR_LEN);
    DR_Money(pp->retailprice);
    DR_VStr(pp->comment);
    DR_End(PART);
    return(0);
}

/*
CREATE TABLE TPCD.PARTSUPP ( PS_PARTKEY    INTEGER NOT NULL,
                              PS_SUPPKEY    INTEGER NOT NULL,
                              PS_AVAILQTY   INTEGER NOT NULL,
                              PS_SUPPLYCOST  DECIMAL(15,2)  NOT
NULL,
                              PS_COMMENT     VARCHAR(199) NOT NULL
);
*/

int
ld_psupp (part_t *pp, int mode)
{
    int i;
    if(mode);
    if (loadinprogress!=PSUPP && loadinprogress!=PART_PSUPP)
start_load(PSUPP,mode);
    for (i = 0; i < SUPP_PER_PART; i++)
    {
        DR_Strt();
        DR_Int(pp->s[i].partkey);

```

```

        DR_Int(pp->s[i].suppkey);
        DR_Int(pp->s[i].qty);
        DR_Money(pp->s[i].scost);
        DR_VStr(pp->s[i].comment);
        DR_End(PSUPP);
    }
    return(0);
}

int
hd_supp (FILE *f)
{
    static int count = 0;
    if(f);

    if (! count++)
        printf("No header has been defined for the supplier
table\n");

    return(0);
}

/*
CREATE TABLE TPCD.SUPPLIER ( S_SUPPKEY    INTEGER NOT NULL,
                             S_NAME       CHAR(25) NOT NULL,
                             S_ADDRESS    VARCHAR(40) NOT NULL,
                             S_NATIONKEY  INTEGER NOT NULL,
                             S_PHONE     CHAR(15) NOT NULL,
                             S_ACCTBAL   DECIMAL(15,2) NOT
NULL,
                             S_COMMENT    VARCHAR(101) NOT
NULL);
*/

int
ld_supp (supplier_t *sp, int mode)
{
    if(mode);
    if (loadinprogress!=SUPP) start_load(SUPP,mode);
    DR_Strt();
    DR_Int(sp->suppkey);
    DR_Str(sp->name,25);
    DR_VStr(sp->address);
    DR_Int(sp->nation_code);
    DR_Str(sp->phone,15);
    DR_Money(sp->acctbal);

        DR_VStr(sp->comment);
        DR_End(SUPP);
    return(0);
}

int
hd_order (FILE *f)
{
    static int count = 0;
    if(f);

    if (! count++)
        printf("No header has been defined for the order
table\n");

    return(0);
}

/*
CREATE TABLE TPCD.ORDER ( O_ORDERKEY    DECIMAL(15,0) NOT
NULL,
                             O_CUSTKEY    INTEGER NOT NULL,
                             O_ORDERSTATUS CHAR(1) NOT NULL,
                             O_TOTALPRICE DECIMAL(15,2) NOT
NULL,
                             O_ORDERDATE  DATE NOT NULL,
                             O_ORDERPRIORITY CHAR(15) NOT NULL,
- R
                             O_CLERK      CHAR(15) NOT NULL,
- R
                             O_SHIPPRIORITY INTEGER NOT NULL,
                             O_COMMENT    VARCHAR(79) NOT
NULL);
*/

int
ld_order (order_t *p, int mode)
{
    if(mode);
    if (loadinprogress!=ORDER && loadinprogress!=ORDER_LINE)
        start_load(ORDER,mode);
    DR_Strt();
    DR_Huge(p->okey); /* even at SF < 300 okey is defined HUGE
*/
    DR_Int(p->custkey);
    DR_Char(p->orderstatus);

```

```

DR_Money(p->totalprice);
DR_Date(p->odate);
DR_Str(p->opriority,O_OPRIO_LEN);
DR_Str(p->clerk,O_CLRK_LEN);
DR_Int(p->spriority);
#ifdef _WIN32
    if (strlen(p->comment)<0||strlen(p->comment)>79)
        __asm int 3;
#endif
DR_VStr(p->comment);
DR_End(ORDER);
return(0);
}

/*
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    DECIMAL(15,2) NOT
NULL,
                             L_EXTENDEDPRICE DECIMAL(15,2) NOT
NULL,
                             L_DISCOUNT   DECIMAL(15,2) NOT
NULL,
                             L_TAX         DECIMAL(15,2) 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, -
- R
                             L_SHIPMODE    CHAR(10) NOT NULL, -
- R
                             L_COMMENT     VARCHAR(44) NOT
NULL);
*/
int
ld_line (order_t *p, int mode)
{
    int i;
    if(mode);
        if (loadinprogress!=LINE && loadinprogress!=ORDER_LINE)
start_load(LINE,mode);
        for (i = 0; i < p->lines; i++)

```

```

{
DR_Strt();
DR_Huge(p->l[i].okey); /* note: SF is not a factor yet */
DR_Int(p->l[i].partkey);
DR_Int(p->l[i].suppkey);
DR_Int(p->l[i].lcnt);
DR_Int(p->l[i].quantity);
DR_Money(p->l[i].eprice);
DR_Money(p->l[i].discount);
DR_Money(p->l[i].tax);
DR_Char(p->l[i].rflag[0]);
DR_Char(p->l[i].lstatus[0]);
DR_Date(p->l[i].sdate);
DR_Date(p->l[i].cdate);
DR_Date(p->l[i].rdate);
DR_Str(p->l[i].shipinstruct, L_INST_LEN);
DR_Str(p->l[i].shipmode, L_SMODE_LEN);
DR_VStr(p->l[i].comment);
DR_End(LINE);
}
return(0);
}

int
hd_psupp (FILE *f)
{
    static int count = 0;
    if(f);

    if (! count++)
        printf("%s %s\n",
                "No header has been defined for the",
                "part supplier table");

return(0);
}

int
hd_line (FILE *f)
{
    static int count = 0;
    if(f);

    if (! count++)

```

```
        printf("No header has been defined for the lineitem
table\n");
```

```
    return(0);
}
```

```
int
hd_nation (FILE *f)
```

```
{
    static int count = 0;
    if(f);
```

```
    if (! count++)
        printf("No header has been defined for the nation
table\n");
```

```
    return(0);
}
```

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

```
int
ld_nation (code_t *cp, int mode)
{
    if(mode);
    if (loadinprogress!=NATION) start_load(NATION,mode);
    DR_Strt();
    DR_Int(cp->code);
    DR_Str(cp->text,NATION_LEN);
    DR_Int(cp->join);
    DR_VStr(cp->comment);
    DR_End(NATION);
    return(0);
}
```

```
int
hd_region (FILE *f)
{
    static int count = 0;
    if(f);

    if (! count++)
```

```
        printf("No header has been defined for the region
table\n");
```

```
    return(0);
}
```

```
/*
CREATE TABLE TPCD.REGION ( R_REGIONKEY INTEGER NOT NULL,
                           R_NAME      CHAR(25) NOT NULL,
                           R_COMMENT   VARCHAR(152));
*/
```

```
int
ld_region (code_t *cp, int mode)
{
    if(mode);
    if (loadinprogress!=REGION) start_load(REGION,mode);
    DR_Strt();
    DR_Int(cp->code);
    DR_Str(cp->text,REGION_LEN);
    DR_VStr(cp->comment);
    DR_End(REGION);

    return(0);
}
```

```
int
ld_order_line (order_t *p, int mode)
{
    if (loadinprogress!=ORDER_LINE)
        start_load(ORDER_LINE,mode);

    ld_order(p, mode);
    ld_line (p, mode);

    return(0);
}
```

```
int
hd_order_line (FILE *f)
{
    hd_order(f);
    hd_line (f);

    return(0);
}
```



```

int
ld_part_psupp (part_t *p, int mode)
{
    if (loadinprogress!=PART_PSUPP)
start_load(PART_PSUPP,mode);
    ld_part(p, mode);
    ld_psupp (p, mode);

    return(0);
}

int
hd_part_psupp (FILE *f)
{
    hd_part(f);
    hd_psupp(f);

    return(0);
}

int
ld_deletes (DSS_HUGE * okey, int mode)
{
    if(mode);
    if (loadinprogress!=19) start_load(19,mode);
    DR_Strt();
    DR_Huge(okey);
    DR_End(19);
    return(0);
}

```

E.3 Load Scripts

Text of Main Load Script

```

-----
#!/bin/sh
#
#
#   Variables describing current system
#
system=local

```

```

SF=`cat SF`
logon="${system}/tpcd${SF}g,tpcd${SF}g"
Logon=".logon $logon"
dbclogon=${system}/dbc,dbc
D=/home2/tpcd
O=${D}/run1
#
#
pernode=2           #number of loaders/node to use
nodes=`egrep -cv '^#|^$' $D/script/LoaderNodes`
children=`expr ${nodes} \* ${pernode}`
#
QUIT=$D/script/quit
#
#
ulimit -n 500
ulimit -f unlimited
ulimit -v unlimited
ulimit -a
#
#
echo "\n-----DBS SETTINGS-----\n"
/tpasw/bin/dbscontrol << INS1
mo in=t
mo ge 13=18
mo fi 1=40
mo fi 3=255
write
di
quit
q
INS1
echo "\n-----CTL SETTINGS-----\n"
/usr/ntos/bin/xctl -nw << INS1
hardware
screen version
screen dbs
screen debug
screen rss
exit
INS1
#
#
#
echo "\n-----\n"
#
#   Start

```

```

#
echo "Running with Scale Factor $SF on `uname -a`, pid=$$"
#
# Setup inmods for current scale factor
#
echo `date +%m/%d/%Y %T " `": Setting up load..."
if [ $SF -gt 300 ]; then
    cp -p $D/pfast/master/MASTER/GT300G/*
    $D/pfast/master/MASTER
else
    cp -p $D/pfast/master/MASTER/LE300G/*
    $D/pfast/master/MASTER
fi
#
mv $D/pfast/master/REMOTES $D/pfast/master/REMOTES.old
echo `date +%m/%d/%Y %T " `": Loading SF=${SF}G with ${nodes}
nodes and ${pernode} loaders per node"
node=0
for i in `egrep -v '^#|^$' $D/script/LoaderNodes`; do
    echo `date +%m/%d/%Y %T " `": Setting up node $i as loader
node ${node}...\c"
    /usr/bin/rsh $i "cd $D/pfast/inmod;./setuplocal ${SF} ${node}
${children} ${pernode}"
    echo "Done"
    echo "REMOTE " $i " root $D/pfast/inmod/{TBL}/{TBL};" >>
    $D/pfast/master/REMOTES
    j=1
    while [ $j -lt $pernode ]; do
        echo "REMOTE " $i " root $D/pfast/inmod/{TBL}${j}/{TBL};" >>
        $D/pfast/master/REMOTES
        j=`expr $j + 1`
    done
    node=`expr $node + 1`
done
#
#####END OF LOAD SETUP#####
#####
#
# Then go and load the tables
#
echo `date +%m/%d/%Y %T " `": Starting Load of Teradata TPC
Benchmark Database..."

cd $D/pfast/master
echo "LOGON $logon;" > LOGON

cd $D/pfast/master

```

```

for i in lineitem order partsupp part supplier customer nation
region; do
    echo `date +%m/%d/%Y %T " `": Starting load of ${i} table"
    ./Build ${i}
    ./fastload < inmod.${i} > inmod.${i}.out 2> inmod.${i}.err
done

cd $D/preps
echo `date +%m/%d/%Y %T " `": Starting base prep...'
echo $Logon | cat - prep_base.sql | bteq > prep_base.out 2>&1
echo `date +%m/%d/%Y %T " `": Starting l_fpk2 prep job...'
echo $Logon | cat - l_fpk2.sql | bteq > l_fpk2.out 2>&1
echo `date +%m/%d/%Y %T " `": Starting o_fpk2 prep job...'
echo $Logon | cat - o_fpk2.sql | bteq > o_fpk2.out 2>&1
echo `date +%m/%d/%Y %T " `": Prep complete, load complete.'

SEED=`date +%m%d%H%M%S`
echo `date +%m/%d/%Y %T " `": preps Complete. SEED will be
$SEED"

cd $D/audit
echo $SEED > SEED_FROM_LOAD
echo `date +%m/%d/%Y %T " `": Running Audit scripts...'
echo ".logon $dbclogon" | cat - space.txt | bteq >
space.${SF}Gload.out 2>space.${SF}Gload.err
echo $Logon | cat - audit.sql | bteq > audit.sql.${SF}Gload.out
2> audit.sql.${SF}Gload.err
echo $Logon | cat - table_contents | bteq >
table_contents.${SF}Gload.out 2> table_contents.${SF}Gload.err
echo `date +%m/%d/%Y %T " `": Audit scripts complete.'

echo `date +%m/%d/%Y %T " `": Exiting'

Text of SF Script:
-----

10000

Text of Build Script:
-----

#!/bin/sh
#
if [ $# != 1 ]; then
    echo "usage: $0 <TPC-D table name>" 1>&2
    exit 1

```

```

fi
tbl=$1
DFLTSESS=1280

if [ \( $tbl = "nation" \) -o \( $tbl = "region" \) ]; then
    sed "s/{TBL}/$1/g" REMOTES | head -1 > R.tmp
    echo "" >> R.tmp
    echo "SESSIONS 1;\n" > S.tmp
else
#
# Get Common REMOTE commands from master directory
# Modify to apply to particular table we are doing
#
    sed "s/{TBL}/${tbl}/g" REMOTES > R.tmp
#
# If master inmod script has a session declaration
# use it, otherwise use default defined above
#
    egrep -i SESSION MASTER/inmod.${tbl} > S.tmp
    if [ $? -ne 0 ]; then
        echo "SESSIONS ${DFLTSESS};\n" > S.tmp
    fi
fi
#
# Put the new script together
# SESSIONS,REMOTE, LOGON,INMOD (sans SESSION)
#
cat S.tmp R.tmp LOGON > inmod.${tbl}
#
# add all of the master script sans any SESSION
# statements
egrep -v -i SESSION MASTER/inmod.${tbl} >> inmod.${tbl}
#
# Finally get rid of the tmp files
rm -f S.tmp R.tmp

```

INMOD FILES CREATED BY THE Build SCRIPT:

```

-----
inmod.lineitem
-----

```

SESSIONS 1280;

```

REMOTE bynetcop1 root
/home2/tpcd/pfast/inmod/lineitem/lineitem;
REMOTE bynetcop1 root
/home2/tpcd/pfast/inmod/lineitem1/lineitem;
REMOTE bynetcop2 root
/home2/tpcd/pfast/inmod/lineitem/lineitem;
REMOTE bynetcop2 root
/home2/tpcd/pfast/inmod/lineitem1/lineitem;
REMOTE bynetcop3 root
/home2/tpcd/pfast/inmod/lineitem/lineitem;
REMOTE bynetcop3 root
/home2/tpcd/pfast/inmod/lineitem1/lineitem;
REMOTE bynetcop4 root
/home2/tpcd/pfast/inmod/lineitem/lineitem;
REMOTE bynetcop4 root
/home2/tpcd/pfast/inmod/lineitem1/lineitem;
REMOTE bynetcop5 root
/home2/tpcd/pfast/inmod/lineitem/lineitem;
REMOTE bynetcop5 root
/home2/tpcd/pfast/inmod/lineitem1/lineitem;
REMOTE bynetcop6 root
/home2/tpcd/pfast/inmod/lineitem/lineitem;
REMOTE bynetcop6 root
/home2/tpcd/pfast/inmod/lineitem1/lineitem;
REMOTE bynetcop7 root
/home2/tpcd/pfast/inmod/lineitem/lineitem;
REMOTE bynetcop7 root
/home2/tpcd/pfast/inmod/lineitem1/lineitem;
REMOTE bynetcop8 root
/home2/tpcd/pfast/inmod/lineitem/lineitem;
REMOTE bynetcop8 root
/home2/tpcd/pfast/inmod/lineitem1/lineitem;
REMOTE bynetcop9 root
/home2/tpcd/pfast/inmod/lineitem/lineitem;
REMOTE bynetcop9 root
/home2/tpcd/pfast/inmod/lineitem1/lineitem;
REMOTE bynetcop10 root
/home2/tpcd/pfast/inmod/lineitem/lineitem;
REMOTE bynetcop10 root
/home2/tpcd/pfast/inmod/lineitem1/lineitem;
REMOTE bynetcop11 root
/home2/tpcd/pfast/inmod/lineitem/lineitem;
REMOTE bynetcop11 root
/home2/tpcd/pfast/inmod/lineitem1/lineitem;
REMOTE bynetcop12 root
/home2/tpcd/pfast/inmod/lineitem/lineitem;

```



```

REMOTE bynetcop58 root
/home2/tpcd/pfast/inmod/lineitem1/lineitem;
REMOTE bynetcop59 root
/home2/tpcd/pfast/inmod/lineitem/lineitem;
REMOTE bynetcop59 root
/home2/tpcd/pfast/inmod/lineitem1/lineitem;
REMOTE bynetcop60 root
/home2/tpcd/pfast/inmod/lineitem/lineitem;
REMOTE bynetcop60 root
/home2/tpcd/pfast/inmod/lineitem1/lineitem;
REMOTE bynetcop61 root
/home2/tpcd/pfast/inmod/lineitem/lineitem;
REMOTE bynetcop61 root
/home2/tpcd/pfast/inmod/lineitem1/lineitem;
REMOTE bynetcop62 root
/home2/tpcd/pfast/inmod/lineitem/lineitem;
REMOTE bynetcop62 root
/home2/tpcd/pfast/inmod/lineitem1/lineitem;
REMOTE bynetcop63 root
/home2/tpcd/pfast/inmod/lineitem/lineitem;
REMOTE bynetcop63 root
/home2/tpcd/pfast/inmod/lineitem1/lineitem;
REMOTE bynetcop64 root
/home2/tpcd/pfast/inmod/lineitem/lineitem;
REMOTE bynetcop64 root
/home2/tpcd/pfast/inmod/lineitem1/lineitem;
LOGON local/tpcd10000g,tpcd10000g;

DROP TABLE LINEITEM;
DROP TABLE ERRLINEITEM1;
DROP TABLE ERRLINEITEM2;

CREATE MULTISET TABLE LINEITEM, DATABLOCKSIZE=112 KILOBYTES
(
  ,L_ORDERKEY          DECIMAL(15,0) not null
  ,L_PARTKEY           INTEGER not null
  ,L_SUPPKEY           INTEGER not null
  ,L_LINENUMBER        INTEGER not null
  ,L_QUANTITY          DECIMAL(15,2) not null
  ,L_EXTENDEDPRICE     DECIMAL(15,2) not null
  ,L_DISCOUNT         DECIMAL(15,2) not null
  ,L_TAX               DECIMAL(15,2) not null
  ,L_RETURNFLAG        CHAR(1) CASESPECIFIC not null
  ,L_LINESTATUS        CHAR(1) CASESPECIFIC not null
  ,L_SHIPDATE          DATE FORMAT 'yyyy-mm-dd' not null
  ,L_COMMITDATE        DATE FORMAT 'yyyy-mm-dd' not null
  ,L_RECEIPTDATE       DATE FORMAT 'yyyy-mm-dd' not null

```

```

  ,L_SHIPINSTRUCT     CHAR(25) CASESPECIFIC not null
  ,L_SHIPMODE          CHAR(10) CASESPECIFIC not null
  ,L_COMMENT           VARCHAR(44) CASESPECIFIC not null)
PRIMARY INDEX (l_orderkey)
PARTITION BY (RANGE_N
              ( L_ShipDate Between
                * AND DATE '1992-12-31',
                DATE '1993-01-01' AND DATE '1993-12-31',
                DATE '1994-01-01' AND DATE '1994-12-31',
                DATE '1995-01-01' AND DATE '1995-12-31',
                DATE '1996-01-01' AND DATE '1996-12-31',
                DATE '1997-01-01' AND DATE '1997-12-31',
                DATE '1998-01-01' AND * ));

BEGIN LOADING LINEITEM
  ERRORFILES ERRLINEITEM1, ERRLINEITEM2
  checkpoint 300;

DEFINE
  ,F_L_ORDERKEY        (DECIMAL(15,0))
  ,F_L_PARTKEY         (INTEGER)
  ,F_L_SUPPKEY         (INTEGER)
  ,F_L_LINENUMBER      (INTEGER)
  ,F_L_QUANTITY        (INTEGER)
  ,F_L_EXTENDEDPRICE   (DECIMAL(15,2))
  ,F_L_DISCOUNT       (DECIMAL(15,2))
  ,F_L_TAX             (DECIMAL(15,2))
  ,F_L_RETURNFLAG      (CHAR(1))
  ,F_L_LINESTATUS      (CHAR(1))
  ,F_L_SHIPDATE        (DATE)
  ,F_L_COMMITDATE      (DATE)
  ,F_L_RECEIPTDATE     (DATE)
  ,F_L_SHIPINSTRUCT    (CHAR(25))
  ,F_L_SHIPMODE        (CHAR(10))
  ,F_L_COMMENT         (VARCHAR(44))

INMOD=BLKEXIT;

SHOW;

INSERT INTO LINEITEM
(
  ,L_ORDERKEY
  ,L_PARTKEY
  ,L_SUPPKEY
  ,L_LINENUMBER
  ,L_QUANTITY
  ,L_EXTENDEDPRICE

```

```

,L_DISCOUNT
,L_TAX
,L_RETURNFLAG
,L_LINESTATUS
,L_SHIPDATE
,L_COMMITDATE
,L_RECEIPTDATE
,L_SHIPINSTRUCT
,L_SHIPMODE
,L_COMMENT
)
VALUES
(
:F_L_ORDERKEY,
:F_L_PARTKEY,
:F_L_SUPPKEY,
:F_L_LINENUMBER,
:F_L_QUANTITY,
:F_L_EXTENDEDPRICE,
:F_L_DISCOUNT,
:F_L_TAX,
:F_L_RETURNFLAG,
:F_L_LINESTATUS,
:F_L_SHIPDATE,
:F_L_COMMITDATE,
:F_L_RECEIPTDATE,
:F_L_SHIPINSTRUCT,
:F_L_SHIPMODE,
:F_L_COMMENT
);

END LOADING;

LOGOFF;

-----
inmod.order
-----

SESSIONS 1280;

REMOTE bynetcop1 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop1 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop2 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop2 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop3 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop3 root /home2/tpcd/pfast/inmod/order1/order;

```

```

REMOTE bynetcop4 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop4 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop5 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop5 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop6 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop6 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop7 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop7 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop8 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop8 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop9 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop9 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop10 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop10 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop11 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop11 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop12 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop12 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop13 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop13 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop14 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop14 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop15 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop15 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop16 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop16 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop17 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop17 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop18 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop18 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop19 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop19 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop20 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop20 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop21 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop21 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop22 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop22 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop23 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop23 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop24 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop24 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop25 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop25 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop26 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop26 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop27 root /home2/tpcd/pfast/inmod/order/order;

```

```

REMOTE bynetcop27 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop28 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop28 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop29 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop29 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop30 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop30 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop31 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop31 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop32 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop32 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop33 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop33 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop34 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop34 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop35 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop35 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop36 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop36 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop37 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop37 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop38 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop38 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop39 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop39 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop40 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop40 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop41 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop41 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop42 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop42 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop43 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop43 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop44 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop44 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop45 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop45 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop46 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop46 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop47 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop47 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop48 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop48 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop49 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop49 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop50 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop50 root /home2/tpcd/pfast/inmod/order1/order;

```

```

REMOTE bynetcop51 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop51 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop52 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop52 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop53 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop53 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop54 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop54 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop55 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop55 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop56 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop56 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop57 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop57 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop58 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop58 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop59 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop59 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop60 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop60 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop61 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop61 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop62 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop62 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop63 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop63 root /home2/tpcd/pfast/inmod/order1/order;
REMOTE bynetcop64 root /home2/tpcd/pfast/inmod/order/order;
REMOTE bynetcop64 root /home2/tpcd/pfast/inmod/order1/order;
LOGON local/tpcd1000g,tpcd1000g;

```

```

DROP TABLE ORDERTBL;
DROP TABLE ERRORORDERTBL1;
DROP TABLE ERRORORDERTBL2;

```

```

CREATE MULTISET TABLE ORDERTBL, DATABLOCKSIZE=112 KILOBYTES
(
    O_ORDERKEY          DECIMAL(15,0) not null
    ,O_CUSTKEY           INTEGER not null
    ,O_ORDERSTATUS      CHAR(1) CASESPECIFIC not null
    ,O_TOTALPRICE       DECIMAL(15,2) not null
    ,O_ORDERDATE        DATE FORMAT 'yyyy-mm-dd' not null
    ,O_ORDERPRIORITY    CHAR(15) CASESPECIFIC not null
    ,O_CLERK            CHAR(15) CASESPECIFIC not null
    ,O_SHIPPRIORITY     INTEGER not null
    ,O_COMMENT          VARCHAR(79) CASESPECIFIC not null
)
PRIMARY INDEX( O_ORDERKEY )

```



```

PARTITION BY (RANGE_N
              ( O_OrderDate Between
                * AND DATE '1992-12-31',
                DATE '1993-01-01' AND DATE '1993-12-31',
                DATE '1994-01-01' AND DATE '1994-12-31',
                DATE '1995-01-01' AND DATE '1995-12-31',
                DATE '1996-01-01' AND DATE '1996-12-31',
                DATE '1997-01-01' AND DATE '1997-12-31',
                DATE '1998-01-01' AND * ));

```

```

BEGIN LOADING ORDERTBL
  ERRORFILES ERRORDERTBL1, ERRORDERTBL2
  checkpoint 300;

```

```

DEFINE
  F_O_ORDERKEY      (DECIMAL(15,0))
  ,F_O_CUSTKEY      (INTEGER)
  ,F_O_ORDERSTATUS  (CHAR(1))
  ,F_O_TOTALPRICE   (DECIMAL(15,2))
  ,F_O_ORDERDATE    (DATE)
  ,F_O_ORDERPRIORITY (CHAR(15))
  ,F_O_CLERK        (CHAR(15))
  ,F_O_SHIPPRIORITY (INTEGER)
  ,F_O_COMMENT      (VARCHAR(79))
INMOD=BLKEXIT;

```

```
SHOW;
```

```

INSERT INTO ORDERTBL
(
  O_ORDERKEY
  ,O_CUSTKEY
  ,O_ORDERSTATUS
  ,O_TOTALPRICE
  ,O_ORDERDATE
  ,O_ORDERPRIORITY
  ,O_CLERK
  ,O_SHIPPRIORITY
  ,O_COMMENT
)

```

```

VALUES
(
  :F_O_ORDERKEY,
  :F_O_CUSTKEY,
  :F_O_ORDERSTATUS,
  :F_O_TOTALPRICE,

```

```

:F_O_ORDERDATE,
:F_O_ORDERPRIORITY,
:F_O_CLERK,
:F_O_SHIPPRIORITY,
:F_O_COMMENT
);

```

```
END LOADING;
```

```
LOGOFF;
```

```
-----
inmod.partsupp
-----
```

```
SESSIONS 1280;
```

```

REMOTE bynetcop1 root
/home2/tpcd/pfast/inmod/partsupp/partsupp;
REMOTE bynetcop1 root
/home2/tpcd/pfast/inmod/partsupp1/partsupp;
REMOTE bynetcop2 root
/home2/tpcd/pfast/inmod/partsupp/partsupp;
REMOTE bynetcop2 root
/home2/tpcd/pfast/inmod/partsupp1/partsupp;
REMOTE bynetcop3 root
/home2/tpcd/pfast/inmod/partsupp/partsupp;
REMOTE bynetcop3 root
/home2/tpcd/pfast/inmod/partsupp1/partsupp;
REMOTE bynetcop4 root
/home2/tpcd/pfast/inmod/partsupp/partsupp;
REMOTE bynetcop4 root
/home2/tpcd/pfast/inmod/partsupp1/partsupp;
REMOTE bynetcop5 root
/home2/tpcd/pfast/inmod/partsupp/partsupp;
REMOTE bynetcop5 root
/home2/tpcd/pfast/inmod/partsupp1/partsupp;
REMOTE bynetcop6 root
/home2/tpcd/pfast/inmod/partsupp/partsupp;
REMOTE bynetcop6 root
/home2/tpcd/pfast/inmod/partsupp1/partsupp;
REMOTE bynetcop7 root
/home2/tpcd/pfast/inmod/partsupp/partsupp;
REMOTE bynetcop7 root
/home2/tpcd/pfast/inmod/partsupp1/partsupp;
REMOTE bynetcop8 root
/home2/tpcd/pfast/inmod/partsupp/partsupp;

```



```

REMOTE bynetcop54 root
/home2/tpcd/pfast/inmod/partsuppl/partsupp;
REMOTE bynetcop55 root
/home2/tpcd/pfast/inmod/partsupp/partsupp;
REMOTE bynetcop55 root
/home2/tpcd/pfast/inmod/partsuppl/partsupp;
REMOTE bynetcop56 root
/home2/tpcd/pfast/inmod/partsupp/partsupp;
REMOTE bynetcop56 root
/home2/tpcd/pfast/inmod/partsuppl/partsupp;
REMOTE bynetcop57 root
/home2/tpcd/pfast/inmod/partsupp/partsupp;
REMOTE bynetcop57 root
/home2/tpcd/pfast/inmod/partsuppl/partsupp;
REMOTE bynetcop58 root
/home2/tpcd/pfast/inmod/partsupp/partsupp;
REMOTE bynetcop58 root
/home2/tpcd/pfast/inmod/partsuppl/partsupp;
REMOTE bynetcop59 root
/home2/tpcd/pfast/inmod/partsupp/partsupp;
REMOTE bynetcop59 root
/home2/tpcd/pfast/inmod/partsuppl/partsupp;
REMOTE bynetcop60 root
/home2/tpcd/pfast/inmod/partsupp/partsupp;
REMOTE bynetcop60 root
/home2/tpcd/pfast/inmod/partsuppl/partsupp;
REMOTE bynetcop61 root
/home2/tpcd/pfast/inmod/partsupp/partsupp;
REMOTE bynetcop61 root
/home2/tpcd/pfast/inmod/partsuppl/partsupp;
REMOTE bynetcop62 root
/home2/tpcd/pfast/inmod/partsupp/partsupp;
REMOTE bynetcop62 root
/home2/tpcd/pfast/inmod/partsuppl/partsupp;
REMOTE bynetcop63 root
/home2/tpcd/pfast/inmod/partsupp/partsupp;
REMOTE bynetcop63 root
/home2/tpcd/pfast/inmod/partsuppl/partsupp;
REMOTE bynetcop64 root
/home2/tpcd/pfast/inmod/partsupp/partsupp;
REMOTE bynetcop64 root
/home2/tpcd/pfast/inmod/partsuppl/partsupp;
LOGON local/tpcd10000g,tpcd10000g;

DROP TABLE PARTSUPP;
DROP TABLE ERRPARTSUPP1;
DROP TABLE ERRPARTSUPP2;

```

```

CREATE MULTISET TABLE PARTSUPP
(
    PS_PARTKEY      INTEGER not null
    ,PS_SUPPKEY     INTEGER not null
    ,PS_AVAILQTY    INTEGER not null
    ,PS_SUPPLYCOST  DECIMAL(15,2) not null
    ,PS_COMMENT     VARCHAR(199) CASESPECIFIC not null
);

BEGIN LOADING PARTSUPP
    ERRORFILES ERRPARTSUPP1, ERRPARTSUPP2
    checkpoint 120;

DEFINE
    F_PS_PARTKEY      (INTEGER)
    ,F_PS_SUPPKEY     (INTEGER)
    ,F_PS_AVAILQTY    (INTEGER)
    ,F_PS_SUPPLYCOST  (DECIMAL(15,2))
    ,F_PS_COMMENT     (VARCHAR(199))
INMOD=BLKEXIT;

SHOW;

INSERT INTO PARTSUPP
(
    PS_PARTKEY
    ,PS_SUPPKEY
    ,PS_AVAILQTY
    ,PS_SUPPLYCOST
    ,PS_COMMENT
)
VALUES
(
    :F_PS_PARTKEY,
    :F_PS_SUPPKEY,
    :F_PS_AVAILQTY,
    :F_PS_SUPPLYCOST,
    :F_PS_COMMENT
);

END LOADING;

LOGOFF;

-----
inmod.part

```



```

REMOTE bynetcop46 root /home2/tpcd/pfast/inmod/part/part;
REMOTE bynetcop46 root /home2/tpcd/pfast/inmod/part1/part;
REMOTE bynetcop47 root /home2/tpcd/pfast/inmod/part/part;
REMOTE bynetcop47 root /home2/tpcd/pfast/inmod/part1/part;
REMOTE bynetcop48 root /home2/tpcd/pfast/inmod/part/part;
REMOTE bynetcop48 root /home2/tpcd/pfast/inmod/part1/part;
REMOTE bynetcop49 root /home2/tpcd/pfast/inmod/part/part;
REMOTE bynetcop49 root /home2/tpcd/pfast/inmod/part1/part;
REMOTE bynetcop50 root /home2/tpcd/pfast/inmod/part/part;
REMOTE bynetcop50 root /home2/tpcd/pfast/inmod/part1/part;
REMOTE bynetcop51 root /home2/tpcd/pfast/inmod/part/part;
REMOTE bynetcop51 root /home2/tpcd/pfast/inmod/part1/part;
REMOTE bynetcop52 root /home2/tpcd/pfast/inmod/part/part;
REMOTE bynetcop52 root /home2/tpcd/pfast/inmod/part1/part;
REMOTE bynetcop53 root /home2/tpcd/pfast/inmod/part/part;
REMOTE bynetcop53 root /home2/tpcd/pfast/inmod/part1/part;
REMOTE bynetcop54 root /home2/tpcd/pfast/inmod/part/part;
REMOTE bynetcop54 root /home2/tpcd/pfast/inmod/part1/part;
REMOTE bynetcop55 root /home2/tpcd/pfast/inmod/part/part;
REMOTE bynetcop55 root /home2/tpcd/pfast/inmod/part1/part;
REMOTE bynetcop56 root /home2/tpcd/pfast/inmod/part/part;
REMOTE bynetcop56 root /home2/tpcd/pfast/inmod/part1/part;
REMOTE bynetcop57 root /home2/tpcd/pfast/inmod/part/part;
REMOTE bynetcop57 root /home2/tpcd/pfast/inmod/part1/part;
REMOTE bynetcop58 root /home2/tpcd/pfast/inmod/part/part;
REMOTE bynetcop58 root /home2/tpcd/pfast/inmod/part1/part;
REMOTE bynetcop59 root /home2/tpcd/pfast/inmod/part/part;
REMOTE bynetcop59 root /home2/tpcd/pfast/inmod/part1/part;
REMOTE bynetcop60 root /home2/tpcd/pfast/inmod/part/part;
REMOTE bynetcop60 root /home2/tpcd/pfast/inmod/part1/part;
REMOTE bynetcop61 root /home2/tpcd/pfast/inmod/part/part;
REMOTE bynetcop61 root /home2/tpcd/pfast/inmod/part1/part;
REMOTE bynetcop62 root /home2/tpcd/pfast/inmod/part/part;
REMOTE bynetcop62 root /home2/tpcd/pfast/inmod/part1/part;
REMOTE bynetcop63 root /home2/tpcd/pfast/inmod/part/part;
REMOTE bynetcop63 root /home2/tpcd/pfast/inmod/part1/part;
REMOTE bynetcop64 root /home2/tpcd/pfast/inmod/part/part;
REMOTE bynetcop64 root /home2/tpcd/pfast/inmod/part1/part;
LOGON local/tpcd10000g,tpcd10000g;

```

```

DROP TABLE PARTTBL;
DROP TABLE ERRPARTTBL1;
DROP TABLE ERRPARTTBL2;

```

```

CREATE MULTISET TABLE PARTTBL
(
    P_PARTKEY          INTEGER not null

```

```

, P_NAME             VARCHAR(55) CASESPECIFIC not null
, P_MFGR             CHAR(25) CASESPECIFIC not null
, P_BRAND            CHAR(10) CASESPECIFIC not null
, P_TYPE             VARCHAR(25) CASESPECIFIC not null
, P_SIZE             INTEGER not null
, P_CONTAINER        CHAR(10) CASESPECIFIC not null
, P_RETAILPRICE      DECIMAL(15,2) not null
, P_COMMENT          VARCHAR(23) CASESPECIFIC not null
)

```

```

UNIQUE PRIMARY INDEX( P_PARTKEY );

```

```

BEGIN LOADING PARTTBL
    ERRORFILES ERRPARTTBL1, ERRPARTTBL2
    checkpoint 60;

```

```

DEFINE
    F_P_PARTKEY      ( INTEGER )
, F_P_NAME          ( VARCHAR(55) )
, F_P_MFGR          ( CHAR(25) )
, F_P_BRAND         ( CHAR(10) )
, F_P_TYPE          ( VARCHAR(25) )
, F_P_SIZE          ( INTEGER )
, F_P_CONTAINER     ( CHAR(10) )
, F_P_RETAILPRICE  ( DECIMAL(15,2) )
, F_P_COMMENT       ( VARCHAR(23) )

```

```

INMOD=BLKEXIT;

```

```

SHOW;

```

```

INSERT INTO PARTTBL

```

```

(
    P_PARTKEY
, P_NAME
, P_MFGR
, P_BRAND
, P_TYPE
, P_SIZE
, P_CONTAINER
, P_RETAILPRICE
, P_COMMENT
)

```

```

VALUES

```

```

(
    :F_P_PARTKEY,
    :F_P_NAME,
    :F_P_MFGR,
    :F_P_BRAND,

```

```

:F_P_TYPE,
:F_P_SIZE,
:F_P_CONTAINER,
:F_P_RETAILPRICE,
:F_P_COMMENT
);

END LOADING;

LOGOFF;

-----
inmod.customer
-----

SESSIONS 1280;

REMOTE bynetcop1 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop1 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop2 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop2 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop3 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop3 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop4 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop4 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop5 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop5 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop6 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop6 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop7 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop7 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop8 root
/home2/tpcd/pfast/inmod/customer/customer;

REMOTE bynetcop8 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop9 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop9 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop10 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop10 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop11 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop11 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop12 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop12 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop13 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop13 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop14 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop14 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop15 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop15 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop16 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop16 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop17 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop17 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop18 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop18 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop19 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop19 root
/home2/tpcd/pfast/inmod/customer1/customer;

```



```

REMOTE bynetcop43 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop43 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop44 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop44 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop45 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop45 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop46 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop46 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop47 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop47 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop48 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop48 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop49 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop49 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop50 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop50 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop51 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop51 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop52 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop52 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop53 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop53 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop54 root
/home2/tpcd/pfast/inmod/customer/customer;

```

```

REMOTE bynetcop54 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop55 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop55 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop56 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop56 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop57 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop57 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop58 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop58 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop59 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop59 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop60 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop60 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop61 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop61 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop62 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop62 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop63 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop63 root
/home2/tpcd/pfast/inmod/customer1/customer;
REMOTE bynetcop64 root
/home2/tpcd/pfast/inmod/customer/customer;
REMOTE bynetcop64 root
/home2/tpcd/pfast/inmod/customer1/customer;
LOGON local/tpcd10000g,tpcd10000g;

DROP TABLE CUSTOMER;
DROP TABLE ERRCUSTOMER1;
DROP TABLE ERRCUSTOMER2;

```

```

CREATE MULTISET TABLE CUSTOMER
(
  C_CUSTKEY      INTEGER not null
  ,C_NAME        VARCHAR(25) CASESPECIFIC not null
  ,C_ADDRESS     VARCHAR(40) CASESPECIFIC not null
  ,C_NATIONKEY   INTEGER not null
  ,C_PHONE       CHAR(15) CASESPECIFIC not null
  ,C_ACCTBAL     DECIMAL(15,2) not null
  ,C_MKTSEGMENT  CHAR(10) CASESPECIFIC not null
  ,C_COMMENT     VARCHAR(117) CASESPECIFIC not null
)
UNIQUE PRIMARY INDEX( C_CUSTKEY );

BEGIN LOADING CUSTOMER
  ERRORFILES ERRCUSTOMER1, ERRCUSTOMER2
  checkpoint 60;

DEFINE
  F_C_CUSTKEY      (INTEGER)
  ,F_C_NAME        (VARCHAR(25))
  ,F_C_ADDRESS     (VARCHAR(40))
  ,F_C_NATIONKEY   (INTEGER)
  ,F_C_PHONE       (CHAR(15))
  ,F_C_ACCTBAL     (DECIMAL(15,2))
  ,F_C_MKTSEGMENT  (CHAR(10))
  ,F_C_COMMENT     (VARCHAR(117))
INMOD=BLKEXIT;

SHOW;

INSERT INTO CUSTOMER
(
  C_CUSTKEY
  ,C_NAME
  ,C_ADDRESS
  ,C_NATIONKEY
  ,C_PHONE
  ,C_ACCTBAL
  ,C_MKTSEGMENT
  ,C_COMMENT
)
VALUES
(
  :F_C_CUSTKEY,
  :F_C_NAME,
  :F_C_ADDRESS,
  :F_C_NATIONKEY,
  :F_C_PHONE,
  :F_C_ACCTBAL,
  :F_C_MKTSEGMENT,
  :F_C_COMMENT
);

END LOADING;

LOGOFF;

-----
inmod.supplier
-----

SESSIONS 1280;

REMOTE bynetcop1 root
/home2/tpcd/pfast/inmod/supplier/supplier;
REMOTE bynetcop1 root
/home2/tpcd/pfast/inmod/supplier1/supplier;
REMOTE bynetcop2 root
/home2/tpcd/pfast/inmod/supplier/supplier;
REMOTE bynetcop2 root
/home2/tpcd/pfast/inmod/supplier1/supplier;
REMOTE bynetcop3 root
/home2/tpcd/pfast/inmod/supplier/supplier;
REMOTE bynetcop3 root
/home2/tpcd/pfast/inmod/supplier1/supplier;
REMOTE bynetcop4 root
/home2/tpcd/pfast/inmod/supplier/supplier;
REMOTE bynetcop4 root
/home2/tpcd/pfast/inmod/supplier1/supplier;
REMOTE bynetcop5 root
/home2/tpcd/pfast/inmod/supplier/supplier;
REMOTE bynetcop5 root
/home2/tpcd/pfast/inmod/supplier1/supplier;
REMOTE bynetcop6 root
/home2/tpcd/pfast/inmod/supplier/supplier;
REMOTE bynetcop6 root
/home2/tpcd/pfast/inmod/supplier1/supplier;
REMOTE bynetcop7 root
/home2/tpcd/pfast/inmod/supplier/supplier;
REMOTE bynetcop7 root
/home2/tpcd/pfast/inmod/supplier1/supplier;
REMOTE bynetcop8 root
/home2/tpcd/pfast/inmod/supplier/supplier;

```



```

REMOTE bynetcop54 root
/home2/tpcd/pfast/inmod/supplier1/supplier;
REMOTE bynetcop55 root
/home2/tpcd/pfast/inmod/supplier/supplier;
REMOTE bynetcop55 root
/home2/tpcd/pfast/inmod/supplier1/supplier;
REMOTE bynetcop56 root
/home2/tpcd/pfast/inmod/supplier/supplier;
REMOTE bynetcop56 root
/home2/tpcd/pfast/inmod/supplier1/supplier;
REMOTE bynetcop57 root
/home2/tpcd/pfast/inmod/supplier/supplier;
REMOTE bynetcop57 root
/home2/tpcd/pfast/inmod/supplier1/supplier;
REMOTE bynetcop58 root
/home2/tpcd/pfast/inmod/supplier/supplier;
REMOTE bynetcop58 root
/home2/tpcd/pfast/inmod/supplier1/supplier;
REMOTE bynetcop59 root
/home2/tpcd/pfast/inmod/supplier/supplier;
REMOTE bynetcop59 root
/home2/tpcd/pfast/inmod/supplier1/supplier;
REMOTE bynetcop60 root
/home2/tpcd/pfast/inmod/supplier/supplier;
REMOTE bynetcop60 root
/home2/tpcd/pfast/inmod/supplier1/supplier;
REMOTE bynetcop61 root
/home2/tpcd/pfast/inmod/supplier/supplier;
REMOTE bynetcop61 root
/home2/tpcd/pfast/inmod/supplier1/supplier;
REMOTE bynetcop62 root
/home2/tpcd/pfast/inmod/supplier/supplier;
REMOTE bynetcop62 root
/home2/tpcd/pfast/inmod/supplier1/supplier;
REMOTE bynetcop63 root
/home2/tpcd/pfast/inmod/supplier/supplier;
REMOTE bynetcop63 root
/home2/tpcd/pfast/inmod/supplier1/supplier;
REMOTE bynetcop64 root
/home2/tpcd/pfast/inmod/supplier/supplier;
REMOTE bynetcop64 root
/home2/tpcd/pfast/inmod/supplier1/supplier;
LOGON local/tpcd10000g,tpcd10000g;

DROP TABLE SUPPLIER;
DROP TABLE ERRSUPPLIER1;
DROP TABLE ERRSUPPLIER2;

```

```

CREATE MULTISET TABLE SUPPLIER
(
    S_SUPPKEY    INTEGER not null
  ,S_NAME       CHAR(25) CASESPECIFIC not null
  ,S_ADDRESS    VARCHAR(40) CASESPECIFIC not null
  ,S_NATIONKEY  INTEGER not null
  ,S_PHONE      CHAR(15) CASESPECIFIC not null
  ,S_ACCTBAL    DECIMAL(15,2) not null
  ,S_COMMENT    VARCHAR(101) CASESPECIFIC not null
)
UNIQUE PRIMARY INDEX( S_SUPPKEY );

BEGIN LOADING SUPPLIER
    ERRORFILES ERRSUPPLIER1, ERRSUPPLIER2
    checkpoint 60;

DEFINE
    F_S_SUPPKEY    (INTEGER)
  ,F_S_NAME       (CHAR(25))
  ,F_S_ADDRESS    (VARCHAR(40))
  ,F_S_NATIONKEY  (INTEGER)
  ,F_S_PHONE      (CHAR(15))
  ,F_S_ACCTBAL    (DECIMAL(15,2))
  ,F_S_COMMENT    (VARCHAR(101))
INMOD=BLKEXIT;

SHOW;

INSERT INTO SUPPLIER
(
    S_SUPPKEY
  ,S_NAME
  ,S_ADDRESS
  ,S_NATIONKEY
  ,S_PHONE
  ,S_ACCTBAL
  ,S_COMMENT
)
VALUES
(
    :F_S_SUPPKEY,
    :F_S_NAME,
    :F_S_ADDRESS,
    :F_S_NATIONKEY,
    :F_S_PHONE,
    :F_S_ACCTBAL,

```

```

        :F_S_COMMENT
    );

END LOADING;

LOGOFF;

-----
inmod.nation
-----

SESSIONS 1;

REMOTE bynetcop1 root /home2/tpcd/pfast/inmod/nation/nation;

LOGON local/tpcd10000g,tpcd10000g;
DROP TABLE NATION;
DROP TABLE ERRNATIONTMP1;
DROP TABLE ERRNATIONTMP2;

CREATE MULTISET TABLE NATION
(
    N_NATIONKEY INTEGER NOT NULL
    ,N_NAME CHAR(25) CASESPECIFIC NOT NULL
    ,N_REGIONKEY INTEGER NOT NULL
    ,N_COMMENT VARCHAR(152) CASESPECIFIC NOT NULL
)
UNIQUE PRIMARY INDEX ( N_NATIONKEY );

BEGIN LOADING NATION
    ERRORFILES ERRNATIONTMP1, ERRNATIONTMP2
    checkpoint 60;

DEFINE
    F_N_NATIONKEY (INTEGER)
    ,F_N_NAME (CHAR(25))
    ,F_N_REGIONKEY (INTEGER)
    ,F_N_COMMENT (VARCHAR(152))
INMOD=BLKEXIT;

SHOW;

INSERT INTO NATION
(
    N_NATIONKEY
    ,N_NAME
    ,N_REGIONKEY

```

```

        ,N_COMMENT
    )
VALUES
(
    :F_N_NATIONKEY
    ,:F_N_NAME
    ,:F_N_REGIONKEY
    ,:F_N_COMMENT
);

END LOADING;

.LOGOFF;

-----
inmod.region
-----

SESSIONS 1;

REMOTE bynetcop1 root /home2/tpcd/pfast/inmod/region/region;

LOGON local/tpcd10000g,tpcd10000g;
DROP TABLE REGION;
DROP TABLE ERRREGIONTMP1;
DROP TABLE ERRREGIONTMP2;

CREATE MULTISET TABLE REGION
(
    R_REGIONKEY INTEGER NOT NULL
    ,R_NAME CHAR(25) CASESPECIFIC NOT NULL
    ,R_COMMENT VARCHAR(152) CASESPECIFIC NOT NULL
)
UNIQUE PRIMARY INDEX ( R_REGIONKEY );

BEGIN LOADING REGION
    ERRORFILES ERRREGIONTMP1, ERRREGIONTMP2
    checkpoint 60;

DEFINE
    F_R_REGIONKEY (INTEGER)
    ,F_R_NAME (CHAR(25))
    ,F_R_COMMENT (VARCHAR(152))
INMOD=BLKEXIT;

SHOW;

```

```
INSERT INTO REGION
(
  R_REGIONKEY
,R_NAME
,R_COMMENT
)
VALUES
(
  :F_R_REGIONKEY
, :F_R_NAME
, :F_R_COMMENT
);

END LOADING;

.LOGOFF;
```

Appendix F. Driver and Implementation Specific Layer Detail

Driver program

The driver program “tpcddriver” was written to simplify running the benchmark. It incorporates all of the QGEN program, plus those parts of the DBGEN program related to generating update sets. This driver is capable of running the entire benchmark, including both the power test and the throughput tests. It uses standard Teradata call-level interface (CLiV2) to send requests to the Teradata system. Because CLiV2 operates at a very low level, an implementation specific layer was added between the driver and CLiV2 to hide some of the complexities of CLiV2 from the main body of the tpcddriver.

The file tpcddriver.c is the main program for the driver. Vsub.c is the same as the standard varsub.c from the standard QGEN code, with modifications to make the substitutions in a buffer instead of writing them to disk. The files build.c, bm_utils.c, rnd.c, and bcd2.c are used unmodified from the standard DBGEN distribution. The include files sqltypes.h and sql.h are standard include files to define X/Open CLI style of interface. These can be found from either the ISO CLI spec (ISO/IEC 9075-3:1995) or from the Microsoft ODBC SDK.

tpcddriver.c

```
/* @(#)tpcddriver.c 2.1.2.2 */
/*
 * tpcddriver.c This program is a master driver program that
 runs
 * the entire TPC-D benchmark. It first runs the power tests,
 then
```

```
 * it runs the throughput tests. It includes the queries and the
 * update functions UF1 and UF2.
 * This program incorporates all of QGEN. It also has some of
 * DBGEN incorporated so it can generate update data on the fly.
 */
#define DECLARER
#define NO_FUNC (int (*) ()) NULL /* to clean up tdefs */
#define NO_LFUNC (long (*) ()) NULL /* to clean up
tdefs */
#define MAXSESS 500
#define MAXKIDS 999
#define EVERYTHING 0x10000
#define LOG0(f, m) \
    if (f & 0x01) { fprintf(fdetail, m); fflush(fdetail); } \
    \
    if (f & 0x02) { fprintf(stdout, m); fflush(fdetail); } \
    if (f & 0x04) fprintf(stderr, m)
#define LOG1(f, m, a1) \
    if (f & 0x01) { fprintf(fdetail, m, a1); fflush(fdetail);
} \
    if (f & 0x02) { fprintf(stdout, m, a1); fflush(fdetail); } \
    \
    if (f & 0x04) fprintf(stderr, m, a1)
#define LOG2(f, m, a1, a2) \
    if (f & 0x01) { fprintf(fdetail, m, a1, a2);
fflush(fdetail); } \
    if (f & 0x02) { fprintf(stdout, m, a1, a2);
fflush(fdetail); } \
    if (f & 0x04) fprintf(stderr, m, a1, a2)
#define LOG3(f, m, a1, a2, a3) \
    if (f & 0x01) { fprintf(fdetail, m, a1, a2, a3);
fflush(fdetail); } \
    if (f & 0x02) { fprintf(stdout, m, a1, a2, a3);
fflush(fdetail); } \
    if (f & 0x04) fprintf(stderr, m, a1, a2, a3)

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <sys/timeb.h>
#include <limits.h>
#include <math.h>
#include <sys/types.h>
#include <sys/timeb.h>
#include <signal.h>
#include <errno.h>
```



```

#if (defined(_POSIX_)||!defined(WIN32))
#include <unistd.h>
#include <sys/wait.h>
#define INLINE _Inline
#else
#include <process.h>
#define INLINE _inline
#endif
#define timeb _timeb
#endif
#define itoa _itoa
#endif
#define ftime _ftime
#endif
#define sleep Sleep(x*1000)
typedef unsigned long DWORD;
__declspec(dllimport) void __stdcall Sleep(DWORD dwMilliseconds);
#endif

#endif
#include <ctype.h>
#include <time.h>
#include <assert.h>

#undef FAR
#define FAR

typedef void *HWND;
#include <sqltypes.h>
#include <sql.h>
#include <sqlext.h>
#include <coptypes.h>
#include <dbcarea.h>
#include <parcel.h>
#undef VERSION

/* includes from QGEN */
#include "config.h"
#include "dss.h"
#include "tpcd.h"
#include "permute.h"
/* These two from DBGEN */
#include "dsstypes.h"
#include "bcd2.h"

/*
 * prototypes for seed functions needed by set_state for insert
 * generation
 * added 13 Aug 99 jms
 */
sd_order (int child, long skip_count);
sd_line (int child, long skip_count);

SQLRETURN SQL_API SQLSetUsingParcel (

        SQLHSTMT hstmt,

        char *uptr,

        long ulen);

SQLRETURN SQL_API SQLWaitForAny (

        SQLHENV henv,

        SQLHSTMT * lphstmt,

        int

        *stmtno);

#endif
typedef int BOOL;
#endif
#define TRUE 1
#endif
#define FALSE 0
#endif
long throughputstreams;
int acl;
char **avl;
int pids[MAXKIDS];
extern seed_t Seed[];

SQLHENV henv;
SQLHDBC hdbcs[MAXSESS + 1];
SQLHSTMT hstmts[MAXSESS + 1];
BOOL hstmtbusy[MAXSESS + 1];
SQLHDBC hdbc;
SQLHSTMT hstmt;

```

```

#define LINE_SIZE 512

/*
 * Function Protoypes
 */

int prep_direct(char *);
int close_direct(void);
int ld_line(order_t * o, int mode);
int ld_order(order_t * o, int mode);
int ld_order_line(order_t * o, int mode);
int ld_deletes (DSS_HUGE * okey, int mode);

void vsub PROTO ((char *SQLRequest, int qnum, int vnum, int
flags));
int strip_comments PROTO ((char *line));
void usage PROTO ((void));
int process_options PROTO ((int cnt, char **args));
int setup PROTO ((void));
void qsub PROTO ((char *qtag, int flags));
void gen_updates (long start, long count, long upd_num);
int gen_deletes (long min, long cnt, long num, int special);

extern char *optarg;
extern int optind;
char **mk_ascdate (void);

char **asc_date;
int snum = -1;
int use_fastload_for_updates = 3;
char *prog;
char *p1 = NULL;
FILE *fdetail;
FILE *fsummary;

int xld_order (order_t * o, int mode);

tdef tdefs[] =
{
    {"part.tbl", "part table", 200000, NO_FUNC,
     {NO_FUNC, NO_FUNC}, NO_LFUNC, NO_FUNC, PSUPP, 0},
    {"partsupp.tbl", "partsupplier table", 200000, NO_FUNC,
     {NO_FUNC, NO_FUNC}, NO_LFUNC, NO_FUNC, NONE, 0},
    {"supplier.tbl", "suppliers table", 10000, NO_FUNC,
     {NO_FUNC, NO_FUNC}, NO_LFUNC, NO_FUNC, NONE, 0},
    {"customer.tbl", "customers table", 150000, NO_FUNC,
     {NO_FUNC, NO_FUNC}, NO_LFUNC, NO_FUNC, NONE, 0},
    {"orders.tbl", "order table", 150000, NO_FUNC,
     {NO_FUNC, NO_FUNC}, sd_order, NO_FUNC, LINE, NONE},
    {"lineitem.tbl", "lineitem table", 150000, NO_FUNC,
     {NO_FUNC, NO_FUNC}, sd_line, NO_FUNC, NONE, 0},
    {"orders.tbl", "order/lineitem tables", 150000, NO_FUNC,
     {NO_FUNC, NO_FUNC}, NO_LFUNC, NO_FUNC, LINE, 0},
    {"part.tbl", "part/partsupplier tables", 200000, NO_FUNC,
     {NO_FUNC, NO_FUNC}, NO_LFUNC, NO_FUNC, PSUPP, 0},
    {"nation.tbl", "nation table", NATIONS_MAX, NO_FUNC,
     {NO_FUNC, NO_FUNC}, NO_LFUNC, NO_FUNC, NONE, NONE},
    {"region.tbl", "region table", NATIONS_MAX, NO_FUNC,
     {NO_FUNC, NO_FUNC}, NO_LFUNC, NO_FUNC, NONE, NONE}
};

long SaveSeed[MAX_STREAM + 1];
long tempSeed;
long initSeeds[MAX_STREAM + 1];
long num_seeds = 0;
long rndm;
int rowcnt = 0;
long minrow = 0, upd_num = 0;
int setsPerReq = 1;
int maxSess = 56;
int specialtest = 0;
double flt_scale;
struct timeb StartTime;
double QueryTime[QUERIES_PER_SET + 1];
double UF1time;
double UF2time;
double Throughputtime;

char SQLRequest[32768];
char UsingBuffer[32768];
void mk_sparse(long i, DSS_HUGE *res, long seq);

/*
 * FUNCTION strip_comments(line)
 *
 * remove all comments from 'line'; recognizes both {} and --
comments
 */

```

```

int
strip_comments (char *line)
{
    static int in_comment = 0;
    char *cp1, *cp2;

    cp1 = line;

#ifdef WIN32
/* Disable warning about conditional expression is constant */
#pragma warning(disable:4127)
#endif
    while (1) /* traverse the entire string
*/
    {
#ifdef WIN32
#pragma warning(default:4127)
#endif
        if (in_comment)
            {
                if ((cp2 = strchr (cp1, '}')) !=
NULL) /* comment ends */
                    {
                        strcpy (cp1, cp2 + 1);
                        in_comment = 0;
                        continue;
                    }
                else
                    {
                        *cp1 = '\0';
                        break;
                    }
            }
        else /* not in_comment */
            {
                if ((cp2 = strchr (cp1, '--')) !=
NULL)
                    {
                        if (*(cp2 + 1) == '-') /*
found a '--' comment */
                            {
                                *cp2 = '\0';
                                break;
                            }
                    }
                if ((cp2 = strchr (cp1, '{')) !=
NULL) /* comment starts */
                    {
                        in_comment = 1;
                        *cp2 = ' ';
                        continue;
                    }
                else
                    break;
            }
    }
    return (0);
}

/*
 * FUNCTION qsub(char *qtag, int flags)
 *
 * based on the settings of flags, and the template file
$QDIR/qtag.sql
 * make the following substitutions to turn a query template into
EQT
 *
 * String      Converted to      Based on
 * =====
 * first line  database <db_name>;  -n from command line
 * second line set explain on;      -x from command line
 * :<number>  parameter <number>
 * :k         set number
 * :o         output to outpath/qnum.snum
 *
 * SET_OUTPUT
 * :s         stream number
 * :b         BEGIN WORK;           -a from command line,
 * START_TRAN
 * :e         COMMIT WORK;         -a from command line,
 * END_TRAN
 * :q         query number
 * :n<number> sets rowcount to be
returned
 */
void
qsub (char *qtag, int flags)
{
    static char *line = NULL, *qpath = NULL;
    int qnum;
    int i;
    FILE *qfp;
    char *cptr, *mark, *qroot = NULL;
    char tempStr[128];

```

```

qnum = atoi (qtag);
if (line == NULL)
    {
        line = malloc (BUFSIZ);
        qpath = malloc (BUFSIZ);
        if (line == NULL || qpath == NULL)
            {
                fprintf (stderr, "Malloc
failed (qsub)\n");
                fflush (stderr);
                fclose (fdetail);
                fclose (fsummary);
                exit (2);
            }
    }

qroot = env_config (QDIR_TAG, QDIR_DFLT);
sprintf (qpath, "%s/%s.sql", qroot, qtag);
qfp = fopen (qpath, "r");
if (qfp == NULL)
    {
        fprintf (stderr, "open failed for query
template %s, errno=%d\n", qpath, errno);
        fflush (stderr);
        fclose (fdetail);
        fclose (fsummary);
        exit (1);
    }

SQLRequest[0] = '\0';
rowcnt = rowcnt_dflt[qnum];
vsub (SQLRequest, qnum, 0, flags); /* set the variables */

/*//if (flags & DFLT_NUM)
//    { strcat(SQLRequest, SET_ROWCOUNT);
strcat(SQLRequest, itoa(rowcnt,tempStr,10)); }
*/
while (fgets (line, BUFSIZ, qfp) != NULL)
    {
        if (line[strlen (line) - 1] == 0x0A) /*
strip LF chars */
            {
                line[strlen (line) - 1] = ' ';
                strcat (line, "\r");
            }
    }

if (line[0] == '.') /* BTEQ command? */
    strcpy (line, " \r");
else if (line[0] == ' ' && line[1] == '.')
    strcpy (line, " \r");

if (!(flags & COMMENT))
    strip_comments (line);

mark = line;
while ((cptr = strchr (mark, VTAG)) != NULL)
    {
        *cptr = '\0';
        cptr++;
        strcat (SQLRequest, mark);
        switch (*cptr)
            {
                case 'b':
                case 'B':
                    if ((flags &
ANSI) == 0)
                        {
                            strcat (SQLRequest, START_TRAN);
                            strcat (SQLRequest, "\r");
                        }
                    cptr++;
                    break;
                case 'c':
                case 'C':
                    /*if (flags &
DBASE)
                        //    {
strcat(SQLRequest, SET_DBASE); strcat(SQLRequest, db_name); }
                    */
                    cptr++;
                    break;
                case 'e':
                case 'E':
                    if ((flags &
ANSI) == 0)
                        {
                            strcat (SQLRequest, END_TRAN);
                            strcat (SQLRequest, "\r");
                        }
                    }
            }
    }

```

```

        }
        cptr++;
        break;
    case 'n':
    case 'N':
        if (!(flags &
DFLT_NUM))
        {
            rowcnt = atoi (++cptr);
            while (isdigit (*cptr) || *cptr == ' ')
                cptr++;
            /*sprintf(tempStr, SET_ROWCOUNT, rowcnt);
//strcat(SQLRequest, tempStr); */
        }
        continue;
    case 'o':
    case 'O':
        if (flags &
OUTPUT)
        {
            /*fprintf(ofp,"%s '%s/%s.%d'", SET_OUTPUT, osuff,
qtag, (snum < 0)?0:snum); */
        }
        cptr++;
        break;
    case 'q':
    case 'Q':
        strcat
(SQLRequest, qtag);
        cptr++;
        break;
    case 's':
    case 'S':
        sprintf (tempStr,
"%d", (snum < 0) ? 0 : snum);
        strcat
(SQLRequest, tempStr);
        cptr++;
        break;
    case 'X':
    case 'x':
        EXPLAIN)
            strcat (SQLRequest, GEN_QUERY_PLAN);
            strcat (SQLRequest, "\r");
            cptr++;
            break;
        default:
            vsub (SQLRequest,
qnum, atoi (cptr), flags & DFLT);
            while (isdigit
(++cptr));
            break;
        }
        mark = cptr;
        strcat (SQLRequest, mark);
    }
    fclose (qfp);
    for (i = strlen (SQLRequest) - 1; i > 0; i--)
        if (SQLRequest[i] == ' ' ||
            SQLRequest[i] == '\r' ||
            SQLRequest[i] == '\t')
            SQLRequest[i] = '\0';
        else
            break;
    if (SQLRequest[strlen (SQLRequest) - 1] == ';')
        SQLRequest[strlen (SQLRequest) - 1] = '\0';
    return;
}

void
usage (void)
{
    printf("TPC-H/R Benchmark Driver, matched to DBGEN/QGEN
(Version %d.%d.%d%c)\n",
VERSION, RELEASE,
MODIFICATION, PATCH);

    printf ("Portions Copyright %s %s\n", TPC, C_DATES);
    printf ("USAGE: %s <options> [ queries ]\n", prog);
    printf ("Options:\n");
}

```

```

    printf ("\t-c\t\t-- retain comments found in
template.\n");
    printf ("\t-d\t\t-- use default substitution values.\n");
    printf ("\t-e\t\t-- run everything: Power test and
throughput test.\n");
    printf ("\t-h\t\t-- print this usage summary.\n");
/*printf("\t-i <str>\t\t-- use file <str> for initialization.\n");
//printf("\t-l <str>\t\t-- log parameters to <str>.\n"); */
    printf ("\t-m <n>\t\t-- number of sessions to use for
updates.\n");
    printf ("\t-M <n>\t\t-- number of transactions per multi-
statement request for updates.\n");
    printf ("\t-n <str>\t\t-- connect to Teradata using logon
<str>.\n");
    printf ("\t-N\t\t-- use default rowcounts and ignore :n
directive.\n");
    printf ("\t-o <str>\t\t-- set the output file base path to
<str>.\n");
    printf ("\t-p <n>\t\t-- use the query permutation for
stream <n>\n");
    printf ("\t-r <n,n,>\t\t-- seed the random number
generator with <n>\n");
    printf ("\t-R \t\t-- seed the random number generator
with the current time of day\n");
    printf ("\t-s <n>\t\t-- base substitutions and updates on
an SF of <n>\n");
    printf ("\t-S <n>\t\t-- set number of streams for
throughput test to <n>\n");
    printf ("\t-U <s>\t\t-- generate <s> update sets. Use 0
to avoid updates\n");
    printf ("\t-C <n>\t\t-- use seed file generated for <n>
load processes to generate updates\n");
    printf ("\t-v\t\t-- verbose.\n");
    printf ("\t-x\t\t-- enable EXPLAIN in each query.\n");
}

int
process_options (int cnt, char **args)
{
    int flag;
    int i;

    while ((flag = getopt (cnt, args,
"acC:dehi:M:m:n:Nl:o:p:P:r:s:S:t:U:vx")) != -1)
        switch (flag)
        {

```

```

case 'a':
/* special test
mode */
        specialtest = 1;
        flags |= VERBOSE;
        break;
case 'c':
/* retain
comments in EQT */
        flags |= COMMENT;
        break;
case 'C':
        children = atoi (optarg);

        break;
case 'd':
/* use default
substitution values */
        flags |= DFLT;
        break;
case 'e':
/* Run
everything, power test + multisession test. */
        snum = 0;
        flags |= EVERYTHING;
        break;
case 'h':
/* just generate
the usage summary */
        usage ();
        exit (0);
        break;
case 'i':
/* set stream
initialization file name */
        ifile = malloc (strlen (optarg) + 1);
        MALLOC_CHECK (ifile);
        strcpy (ifile, optarg);
        flags |= INIT;
        break;
case 'l':
/* log parameter
usages */
        lfile = malloc (strlen (optarg) + 1);
        MALLOC_CHECK (lfile);
        strcpy (lfile, optarg);
        flags |= LOG;
        break;
case 'm':
/* sessions to
use for updates. */
        maxSess = atoi (optarg);
        if (maxSess < 1 || maxSess > MAXSESS)
        {

```

```

        fprintf (stderr, " Max
sessions out of range, ignored\n");
        maxSess = 56;
    }
    break;

    case 'M':
        /* transactions
per multi-statement req. */
        setsPerReq = atoi (optarg);
        if (setsPerReq < 1 || setsPerReq > 9)
        {
            fprintf (stderr, " Sets per
Request out of range, ignored\n");
            setsPerReq = 1;
        }
        break;

    case 'N':
        /* use default
rowcounts */
        flags |= DFLT_NUM;
        break;

    case 'n':
        /* set database
name */
        db_name = malloc (strlen (optarg) +
1);
        MALLOC_CHECK (db_name);
        strcpy (db_name, optarg);
        flags |= DBASE;
        break;

    case 'o':
        /* set the output
path */
        osuff = malloc (strlen (optarg) + 1);
        MALLOC_CHECK (osuff);
        strcpy (osuff, optarg);
        flags |= OUTPUT;
        break;

    case 'p':
        /* permutation
for a given stream */
        snum = atoi (optarg);
        break;

    case 'P':
        use_fastload_for_updates = atoi
(optarg);
        break;

    case 'r':
        /* set random
number seed for parameter gen */
        flags |= SEED;

```

```

        num_seeds = sscanf (optarg,
"%ld,%ld,%ld,%ld,%ld,%ld,%ld,%ld,%ld,%ld,%ld",
        &initSeeds[0], &initSeeds[1], &initSeeds[2],
&initSeeds[3], &initSeeds[4], &initSeeds[5],
        &initSeeds[6], &initSeeds[7], &initSeeds[8],
&initSeeds[9], &initSeeds[10], &initSeeds[11]);
        rndm = initSeeds[0];
        if (num_seeds > 1)
        {
            for (i = 0; i <
num_seeds; i++)
            {
                //printf
(" -- Using %ld as seed for stream %d\n", initSeeds[i], i);
                if
                {
                    fprintf (stderr, " -- Seeds cannot be zero!\n");
                    exit (1);
                }
            }
        }
        //else
        // printf (" -- Using %ld as
seed\n", rndm);
        break;

    case 's':
        /* scale of data
set to run against */
        flt_scale = (double) atof (optarg);
        if (flt_scale < 1.0)
        {
            int i;
            scale = 1;
            for (i = PART; i < LINE;
            {
                tdefs[i].base = (long) (tdefs[i].base * flt_scale);
                if
                (tdefs[i].base < 1)

```

```

tdefs[i].base = 1;
    }
    else
        scale = (long) flt_scale;
    if (scale < 1)
    {
        fprintf (stderr,
                "WARNING: Scale
set to its lower bound (1)\n");
        scale = 1;
    }
    break;

case 'S':
    throughputstreams = atoi (optarg);
    if (throughputstreams > MAX_STREAM)
    {
        fprintf (stderr,
"Throughput streams reset to limit of %d\n",
MAX_STREAM);
        throughputstreams =
MAX_STREAM;
    }
    break;
case 't':
    /* set
termination file name */
    tfile = malloc (strlen (optarg) + 1);
    MALLOC_CHECK (tfile);
    strcpy (tfile, optarg);
    flags |= TERMINATE;
    break;
case 'U':
    /* generate for
update stream */
    updates = atoi (optarg);
    break;
case 'v':
    /* verbose */
    flags |= VERBOSE;
    break;
case 'x':
    /* set explain in
the queries */
    flags |= EXPLAIN;
    break;
default:

```

```

    printf ("unknown option '%s'
ignored\n", args[optind]);
    usage ();
    exit (1);
    break;
    }
    if (children > MAXKIDS && step < 0)
        /*
limitation of current seed file names */
    {
        printf ("Child process counts of > %d not
supported.\n", MAXKIDS);
        exit (1);
    }
    /*
    if (children > 1 && step >= 0)
        pids = malloc (children * sizeof (pid_t)); */
    return (0);
}

int
setup (void)
{
    asc_date = mk_ascdate ();
    read_dist (env_config (DIST_TAG, DIST_DFLT), "p_cntr",
&p_cntr_set);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "colors",
&colors);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "p_types",
&p_types_set);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "nations",
&nations);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "nations2",
&nations2);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "regions",
&regions);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "o_oprio",
&o_priority_set);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "instruct",
&l_instruct_set);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "smode",
&l_smode_set);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "category",
&l_category_set);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "rflag",
&l_rflag_set);
    read_dist (env_config (DIST_TAG, DIST_DFLT), "msegmnt",
&c_mseg_set);

```



```

        /* load the distributions that contain text generation */
        read_dist (env_config (DIST_TAG, DIST_DFLT), "nouns",
&nouns);
        read_dist (env_config (DIST_TAG, DIST_DFLT), "verbs",
&verbs);
        read_dist (env_config (DIST_TAG, DIST_DFLT), "adjectives",
&adjectives);
        read_dist (env_config (DIST_TAG, DIST_DFLT), "adverbs",
&adverbs);
        read_dist (env_config (DIST_TAG, DIST_DFLT), "auxillaries",
&auxillaries);
        read_dist (env_config (DIST_TAG, DIST_DFLT), "terminators",
&terminators);
        read_dist (env_config (DIST_TAG, DIST_DFLT), "articles",
&articles);
        read_dist (env_config (DIST_TAG, DIST_DFLT), "prepositions",
&prepositions);
        read_dist (env_config (DIST_TAG, DIST_DFLT), "grammar",
&grammar);
        read_dist (env_config (DIST_TAG, DIST_DFLT), "np", &np);
        read_dist (env_config (DIST_TAG, DIST_DFLT), "vp", &vp);
        read_dist (env_config (DIST_TAG, DIST_DFLT), "q13a", &q13a);
        read_dist (env_config (DIST_TAG, DIST_DFLT), "q13b", &q13b);

        return (0);
}

BOOL canRetry(long ErrorCode)
{
    if (ErrorCode == 2825 || /* No record of transaction after
DBC restart */
        ErrorCode == 2826 || /* Request complete, but
output lost due to DBC Restart */
        ErrorCode == 2584 || /* Transaction aborted due to
Disk I/O error */
        ErrorCode == 2631 || /* Transaction aborted due to
deadlock */
        ErrorCode == 3120 || /* Request lost due to DBC
recovery */
        ErrorCode == 3679 || /* Transaction aborted due to
resource shortage */
        ErrorCode == 3897 /* Transaction aborted due to
DBC restart */ )
        return TRUE;
    else
        return FALSE;
}

```

```

/* First, some helper routines to copy data into the parcel we
are building */
static void
DR_Strt ()
{
    pl = UsingBuffer; /* Skip 2 for the length
field */
}
static void
DR_End (int t)
{
    if (t);
    if ((pl - UsingBuffer) > 20000)
        fprintf (stderr, " Using data too long!!!!!!\n");
    if ((pl - UsingBuffer) + strlen (SQLRequest) > 32000)
        fprintf (stderr, " Using data too long!!!!!!\n");
    SQLSetUsingParcel (hstmt, UsingBuffer, (int) (pl -
UsingBuffer));
}

INLINE
static void
DR_Int (long x)
{
    memcpy (pl, &x, 4);
    pl += 4;
}

INLINE
static void DR_Huge(DSS_HUGE * data) /* Used only for orderkey,
which gets > 32 bits at times */
{
#ifdef SUPPORT_64BITS
    if (scale > 300) {
        memcpy(pl,data,8);
        pl+=8;
    }
    else {
        memcpy(pl,data,4);
        pl+=4;
    }
}
#else
    double tempx;
    unsigned long templ;
    long temph;

```

```

    if (*(data+1) ==0)
    {
        memcpy(pl,*data,4);
        pl += 4;
        memcpy(pl,*(data+1),4);
        pl += 4;
    }
    else
    {
        /* Sigh, don't have an Int64 type, so need
to use double and convert to 2 longs */
        tempx = *(data+1)*10000000.0+*data;
        tempy = (long)(tempx/4294967296.0);
        templ = (unsigned long)(tempx-
tempy*4294967296.0);
        memcpy(pl,&templ,4);
        pl += 4;
        memcpy(pl,&temph,4);
        pl += 4;
    }
#endif
}
INLINE
static void
DR_VStr (char *x)
{
    short len;
    len = (short) strlen (x);
    memcpy (pl, &len, 2);
    pl += 2;
    memcpy (pl, x, len);
    pl += len;
}
INLINE
static void
DR_Str (char *x, size_t y)
{
    memset (pl, ' ', y);
    memcpy (pl, x, strlen (x));
    pl += y;
}
INLINE
static void
DR_Money (long x)
{
    long temp = 0;

```

```

        memcpy (pl, &x, 4);
        pl += 4;
        memcpy (pl, &temp, 4);
        pl += 4;
    }
    INLINE
    static void
    DR_Char (char x)
    {
        *pl = x;
        pl++;
    }
    INLINE
    static void
    DR_Date (char *x)
    {
        long tempdate;
        int y, m, d;
        y = atoi (x);
        m = atoi (x + 5);
        d = atoi (x + 8);
        if (y < 1800 || y > 2200 || m < 1 || m > 12 || d < 1 || d
> 31)
        {
            fprintf (stderr, " Help! I can't interpret
the date %s\n", x);
            exit (1);
        }
        tempdate = (y - 1900) * 10000 + m * 100 + d;
        memcpy (pl, &tempdate, 4);
        pl += 4;
    }
}
/*
CREATE TABLE TPCD.ORDER ( O_ORDERKEY          DECIMAL(15,0) NOT
NULL,
O_CUSTKEY          INTEGER NOT NULL,
O_ORDERSTATUS     CHAR(1) NOT NULL,
O_TOTALPRICE      DECIMAL(15,2) NOT NULL,
O_ORDERDATE       DATE NOT NULL,
O_ORDERPRIORITY   CHAR(15) NOT NULL, -- R
O_CLERK           CHAR(15) NOT NULL, -- R
O_SHIPPRIORITY    INTEGER NOT NULL,
O_COMMENT         VARCHAR(79) NOT NULL);
*/

```

```

int
xld_order (order_t * p, int mode)
{
    if (mode);

    DR_Huge (p->okey);
    DR_Int (p->custkey);
    DR_Char (p->orderstatus);
    DR_Money (p->totalprice);
    DR_Date (p->odate);
    DR_Str (p->opriority, O_OPPIO_LEN);
    DR_Str (p->clerk, O_CLRK_LEN);
    DR_Int (p->spriority);
    DR_VStr (p->comment);
    DR_End (ORDER);
    return (0);
}

/*
CREATE TABLE TPCD.LINEITEM ( L_ORDERKEY      DECIMAL(15,0) NOT
NULL,
L_PARTKEY      INTEGER NOT NULL,
L_SUPPKEY      INTEGER NOT NULL,
L_LINENUMBER   INTEGER NOT NULL,
L_QUANTITY     DECIMAL(15,2) NOT NULL,
L_EXTENDEDPRICE DECIMAL(15,2) NOT NULL,
L_DISCOUNT    DECIMAL(15,2) NOT NULL,
L_TAX          DECIMAL(15,2) 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, -- R
L_SHIPMODE     CHAR(10) NOT NULL, -- R
L_COMMENT      VARCHAR(44) NOT NULL);
*/
xld_one_line (order_t * p, int i, int mode)
{
    //int i;
    if (mode);
    //for (i = 0; i < p->lines; i++)
    //    {
int
xld_order (order_t * p, int mode)
{
    DR_Huge (p->l[i].okey);
    DR_Int (p->l[i].partkey);
    DR_Int (p->l[i].suppkey);
    DR_Int (p->l[i].lcnt);
    DR_Int (p->l[i].quantity);
    DR_Money (p->l[i].eprice);
    DR_Money (p->l[i].discount);
    DR_Money (p->l[i].tax);
    DR_Char (p->l[i].rflag[0]);
    DR_Char (p->l[i].lstatus[0]);
    DR_Date (p->l[i].sdate);
    DR_Date (p->l[i].cdate);
    DR_Date (p->l[i].rdate);
    DR_Str (p->l[i].shipinstruct, L_INST_LEN);
    DR_Str (p->l[i].shipmode, L_SMODE_LEN);
    DR_VStr (p->l[i].comment);
    DR_End (LINE);
}
//    }
return (0);
}

/*
int
xld_line (order_t * p, int mode)
{
    int i;
    xld_order (p, mode);
    for (i = 0; i < p->lines; i++)
        xld_one_line (p, i, mode);
    return (0);
}

int
xld_order_line (order_t * p, int mode)
{
    xld_order (p, mode);
    xld_line (p, mode);

    return (0);
} */

int
Fetch_and_Display_All_Rows (SQLHSTMT hstmt, int qnum)
{

```

```

int rc;
int rc2;
char dValue[512];
char ColTitle[60];
short ColTitleLen;
long actualLen;
short icol;
short maxcol;
long totalRows;
long columnDisplaySize[200];
long columnSQLType[200];
long currow;
long temp;
char SqlState[6];
char ErrMsg[256];
short ErrMsgLen;
long ErrCode;

if (SQLRowCount (hstmt, &totalRows) != SQL_SUCCESS)
    fprintf (stderr, " SQLRowCount problem\n");
if (SQLNumResultCols (hstmt, &maxcol) != SQL_SUCCESS)
    fprintf (stderr, " SQLNumResultCols problem\n");
if (maxcol == 0)
    {
        LOG0(3, "\n SQL statement complete.\n");
    }
else
    {
        if (qnum > 0)
            {
                LOG2(3,
                    "\n Query %d complete, %d rows
returned\n", qnum, totalRows);
            }
        else
            {
                LOG1(3,
                    " Query complete, %d rows
returned\n", totalRows);
            }

        if (!(flags & EXPLAIN))
            if (rowcnt > 0 && rowcnt < totalRows)
                {
                    LOG1(1, " only
displaying %d rows per TPC-D spec\n", rowcnt);
                }
    }

```

```

totalRows = rowcnt;
    }
    LOG0(1, "\n\n");
}
currow = 0;
actualLen = 0;
if (totalRows != 0 && maxcol > 0)
    {
        for (icol = 1; icol <= maxcol; icol++)
            {
                ColTitle[0] = '\0';
                if (SQLColAttributes (hstmt,
                    icol, SQL_COLUMN_LABEL, ColTitle, 60, &ColTitleLen, &temp) !=
                    SQL_SUCCESS)
                    {
                        SQLError
                        ((SQLHENV) 0, (SQLHDBC) 0, hstmt, (SQLCHAR *) SqlState, &ErrCode,
                        (SQLCHAR *) ErrMsg, 256, &ErrMsgLen);

                        fprintf (stderr,
                            " ColAttributes call failed\n");
                        fprintf (stderr,
                            " %s (%ld) %s\n", SqlState, ErrCode, ErrMsg);
                    }
                if (SQLColAttributes (hstmt,
                    icol, SQL_COLUMN_DISPLAY_SIZE, NULL, 0, NULL,
                    &columnDisplaySize[icol]) != SQL_SUCCESS)
                    {
                        SQLError
                        ((SQLHENV) 0, (SQLHDBC) 0, hstmt, (SQLCHAR *) SqlState, &ErrCode,
                        (SQLCHAR *) ErrMsg, 256, &ErrMsgLen);

                        fprintf (stderr,
                            " ColAttributes call failed\n");
                        fprintf (stderr,
                            " %s (%ld) %s\n", SqlState, ErrCode, ErrMsg);
                    }
                if (SQLColAttributes (hstmt,
                    icol, SQL_COLUMN_TYPE, NULL, 0, NULL, &columnSQLType[icol]) !=
                    SQL_SUCCESS)
                    {
                        SQLError
                        ((SQLHENV) 0, (SQLHDBC) 0, hstmt, (SQLCHAR *) SqlState, &ErrCode,
                        (SQLCHAR *) ErrMsg, 256, &ErrMsgLen);

                        fprintf (stderr,
                            " ColAttributes call failed\n");
                    }
            }
    }

```

```

        fprintf (stderr,
" %s (%ld) %s\n", SqlState, ErrCode, ErrMsg);
    }
    LOGl(1, "%c", '|');
    actualLen = strlen (ColTitle);
    if (columnDisplaySize[icol] <
actualLen)
        columnDisplaySize[icol]
= actualLen;

    if (columnSQLType[icol] !=
SQL_CHAR && columnSQLType[icol] != SQL_VARCHAR)
        while
(columnDisplaySize[icol] > actualLen++ && actualLen < 81)
            {
                LOGl(1, "%c",
'_');
            }

    LOGl(1, "%s", ColTitle);

    if (columnSQLType[icol] ==
SQL_CHAR || columnSQLType[icol] == SQL_VARCHAR)
        while
(columnDisplaySize[icol] > actualLen++ && actualLen < 81)
            {
                LOGl(1, "%c",
'_');
            }
        }
    LOG0(1, "|\\n");
}

rc = SQL_SUCCESS;
if (totalRows > 0)
    while (((rc = SQLFetch (hstmt)) == SQL_SUCCESS) &&
currow++ < totalRows) /* records */
    {
        icol = 0;
        actualLen = 0;
        rc2 = SQL_SUCCESS;
        while (((int) icol++ <= (int) maxcol)
&&
                (rc2 = SQLGetData (hstmt,
icol, SQL_C_CHAR, dValue, 512, &actualLen) == SQL_SUCCESS))
            {
                LOGl(1, "%c", '|');
                if (columnSQLType[icol]
!= SQL_CHAR && columnSQLType[icol] != SQL_VARCHAR)
                    while
(columnDisplaySize[icol] > actualLen++ && actualLen < 81)
                        {
                            LOGl(1,
"%c", ' ');
                        }
                    LOGl(1, "%s", dValue);
                if (columnSQLType[icol]
== SQL_CHAR || columnSQLType[icol] == SQL_VARCHAR)
                    while
(columnDisplaySize[icol] > actualLen++ && actualLen < 81)
                        {
                            LOGl(1,
"%c", ' ');
                        }
                }
                LOG0(1, "|\\n");
                if (rc2 == SQL_ERROR)
                    {
                        SQLError ((SQLHENV) 0,
(SQLHDBC) 0, hstmt, (SQLCHAR *) SqlState, &ErrCode, (SQLCHAR *)
ErrMsg, 256, &ErrMsgLen);

                        fprintf (stderr, "
GetData call failed\\n");

                        fprintf (stderr, " %s
(%ld) %s\n", SqlState, ErrCode, ErrMsg);
                    }
                }
                LOG0(1, "\\n");
                if (rc == SQL_ERROR)
                    {
                        SQLError ((SQLHENV) 0, (SQLHDBC) 0, hstmt,
(SQLCHAR *) SqlState, &ErrCode, (SQLCHAR *) ErrMsg, 256,
&ErrMsgLen);

                        fprintf (stderr, " Fetch call failed\\n");
                        fprintf (stderr, " %s (%ld) %s\n", SqlState,
ErrCode, ErrMsg);
                    }
            }
    }
}

```

```

    if (rc == SQL_ERROR)
        return ((SQLRETURN) rc);

    return (SQLFreeStmt (hstmt, SQL_CLOSE));
}

```

```

DSS_HUGE * start;
DSS_HUGE * last;

```

```

long orderRowsDeleted = 0;
long lineitemRowsDeleted = 0;
long orderRowsToBeDeleted = 0;
long orderRowsInError = 0;
int OrdersInReq[MAXSESS + 1] = {0};

```

```

typedef struct DBC
{
    Int32 SessionId;
    Int32 ReqId;
    short numResultCols;
    long RowCount;
    char *p;
    unsigned short Async;
    unsigned short busy;
    long NativeError;
    char ErrorMessage[512];
    struct CliDataInfoType DataInfo;
    struct DBCAREA dbcarea;
}

```

```

DBC;

```

```

typedef struct STMT
{
    DBC *lpdbc;
    long hstmt_num;
}

```

```

STMT;

```

```

SQLRETURN
Get_Delete_Answer (int h, SQLRETURN rc)
{
    long totalRows;

```

```

char SqlState[6];
char ErrMsg[256];
short ErrMsgLen;
long ErrCode;
long j;

```

```

    if (rc == SQL_ERROR)
    {
        SQLError ((SQLHENV) 0, (SQLHDBC) 0,
            hstmts[h], (SQLCHAR *) SqlState, &ErrCode, (SQLCHAR *) ErrMsg,
            256, &ErrMsgLen);

        if (canRetry(ErrCode)) /* Transaction
            ABORTed due to DeadLock */
        {
            LOG0(3, " Deadlock during
                delete\n");
            LOG3(3, " %s (%ld) %s\n",
                SqlState, ErrCode, ErrMsg);
            SQLFreeStmt (hstmts[h],
                SQL_CLOSE);
            LOG0(3, " Attempting to
                resubmit\n");
            rc = SQLExecDirect(hstmts[h],
                (SQLCHAR FAR *)((STMT *)
                    hstmt)->lpdbc->dbcarea.req_ptr,
                (SQLINTEGER) ((STMT *)
                    hstmt)->lpdbc->dbcarea.req_len);
            if (rc == SQL_ERROR)
            {
                LOG0(3, " Resubmit
                    failed to work\n");
            }
            else
                return
                    Get_Delete_Answer(h,rc);
        }
    }

```

```

        orderRowsInError += OrdersInReq[h];
        hstmtbusy[h] = FALSE;
        DR_Strt ();

```

```

        LOG0(3, " Delete Request failed\n");
        LOG1(3, "%s\n", SQLRequest);
        LOG3(3, " %s (%ld) %s\n", SqlState, ErrCode,
            ErrMsg);

```

```

        SQLFreeStmt (hstmts[h], SQL_CLOSE);
        fprintf (stderr, " Delete request failed,
Cannot continue!\n");
        DR_Strt ();
        return (-1);
    }
else
    {
        DR_Strt ();
        if (flags & EXPLAIN)
            {
                hstmtbusy[h] = FALSE;
                SQLSetStmtOption (hstmts[h],
SQL_ASYNC_ENABLE, SQL_ASYNC_ENABLE_OFF);
                LOG1(1, "%s\n\n", SQLRequest);
                Fetch_and_Display_All_Rows
(hstmts[h], -1);
                return (-1);
            }
        else
            {
                long templong;

                hstmtbusy[h] = FALSE;
                SQLSetStmtOption (hstmts[h],
SQL_ASYNC_ENABLE, SQL_ASYNC_ENABLE_OFF);

                for (j = 1; j <=
OrdersInReq[h]; j++)
                    {
                        SQLRowCount
(hstmts[h], &totalRows);

                        orderRowsDeleted
+= totalRows;

                        if (totalRows !=
1 && totalRows != 0)
                            {
                                LOG1(1, "
%d rows deleted from Order Table\n", totalRows);
                                }
                            if (j !=
OrdersInReq[h])

```

```

            if ((rc =
SQLMoreResults (hstmts[h])) != SQL_SUCCESS)
                {
                    fprintf (stderr, " Problem with SQLMoreResult in del \n");

                    if (rc == SQL_NO_DATA_FOUND)

                        fprintf (stderr, " No data found from
SQLMoreResults in Del\n");

                    else
                        {
                            SQLError ((SQLHENV) 0, (SQLHDBC) 0,
hstmts[h], (SQLCHAR *) SqlState, &ErrCode, (SQLCHAR *) ErrMsg,
256, &ErrMsgLen);

                            fprintf (stderr, " %s (%ld) %s\n", SqlState,
ErrCode, ErrMsg);

                            LOG0(5, " Something is wrong in the delete
response\n");
                        }
                    break;
                }
            }
        }
        //fprintf(stdout, " %d rows
deleted from Order Table\n",totalRows);
        totalRows = 0;
        while ((rc = SQLMoreResults
(hstmts[h])) == SQL_SUCCESS)
            {
                templong = 0;
                SQLRowCount
(hstmts[h], &templong);

                if (templong < 0
|| templong > 10)
                    {
                        LOG1(1, "
%d rows deleted from LineItem table\n", templong);
                    }
            }

```

```

    }
    totalRows +=
templong;
    }
    if (rc == SQL_ERROR)
    {
        fprintf (stderr,
" Problem with SQLMoreResult del2\n");
        SQLERROR
((SQLHENV) 0, (SQLHDBC) 0, hstmts[h], (SQLCHAR *) SqlState,
&ErrCode, (SQLCHAR *) ErrMsg, 256, &ErrMsgLen);
        fprintf (stderr,
" %s (%ld) %s\n", SqlState, ErrCode, ErrMsg);
    }
    lineitemRowsDeleted +=
totalRows;
}
}
return 0;
}

int LinesPerOrder[MAXSESS + 1][20];
long orderRowsAdded = 0;
long lineitemRowsAdded = 0;
long orderRowsToBeAdded = 0;
long lineitemRowsToBeAdded = 0;
long duplicates = 0;
order_t o[MAXSESS + 1][20];

SQLRETURN
Get_Insert_Answer (int h, SQLRETURN rc)
{
    long totalRows;
    char SqlState[6];
    char ErrMsg[256];
    short ErrMsgLen;
    long ErrCode;
    int j;

    if (rc == SQL_ERROR)
    {
        hstmtbusy[h] = FALSE;
        SQLERROR ((SQLHENV) 0, (SQLHDBC) 0,
hstmts[h], (SQLCHAR *) SqlState, &ErrCode, (SQLCHAR *) ErrMsg,
256, &ErrMsgLen);
        if (ErrCode == 2801) /* Duplicate unique
primary index */
        {
            DR_Strt ();
            duplicates += OrdersInReq[h];
            LOG1(3, " Duplicate on
orderkey %d\n", *(o[h][0].okey));
            LOG3(3, " %s (%ld) %s\n",
SqlState, ErrCode, ErrMsg);
            SQLFreeStmt (hstmts[h],
SQL_CLOSE);
        }
        else if (canRetry(ErrCode)) /* Transaction
ABORTed due to DeadLock */
        {
            LOG1(3, " Deadlock on orderkey
%d\n", *(o[h][0].okey));
            LOG3(3, " %s (%ld) %s\n",
SqlState, ErrCode, ErrMsg);
            SQLFreeStmt (hstmts[h],
SQL_CLOSE);
            LOG0(3, " Attempting to
resubmit\n");
            rc = SQLExecDirect(hstmts[h],
(SQLCHAR FAR *)((STMT *)
hstmt)->lpdbc->dbcarea.req_ptr,
(SQLINTEGER) ((STMT *)
hstmt)->lpdbc->dbcarea.req_len);
            if (rc == SQL_ERROR)
            {
                LOG0(3, " Resubmit
failed to work\n");
            }
            else
                return
Get_Insert_Answer(h,rc);
        }
        else
        {
            DR_Strt ();

```



```

        orderRowsInError +=
OrdersInReq[h];
failed\n");
SqlState, ErrCode, ErrMsg);
SQL_CLOSE);
request failed, Cannot continue!\n");
        DR_Strt ();
        OrdersInReq[h] = 0;
        return (-1);
    }
    OrdersInReq[h] = 0;
else
    {
        DR_Strt ();
        if (flags & EXPLAIN)
            {
                hstmtbusy[h] = FALSE;
                SQLSetStmtOption (hstmts[h],
SQL_ASYNC_ENABLE, SQL_ASYNC_ENABLE_OFF);
                OrdersInReq[h] = 0;
                LOG1(1, "%s\n\n", SQLRequest);
                Fetch_and_Display_All_Rows
(hstmts[h], -1);
                return (-1);
            }
        else
            {
                long templong;
                int k;

                hstmtbusy[h] = FALSE;
                SQLSetStmtOption (hstmts[h],
SQL_ASYNC_ENABLE, SQL_ASYNC_ENABLE_OFF);

                for (j = 0; j <
OrdersInReq[h]; j++)
                    {
                        SQLRowCount
(hstmts[h], &totalRows);
                        orderRowsAdded +=
totalRows;
                    }
                if (totalRows !=
1)
                    {
                        LOG1(1, "
%d rows inserted into Order Table in this request! \n",
totalRows);
                    }
                totalRows = 0;
                for (k = 0; k <
LinesPerOrder[h][j]; k++)
                    {
                        if
(SQLMoreResults (hstmts[h]) != SQL_SUCCESS)
                            {
                                fprintf (stderr, " Problem with SQLMoreResult for
insert \n");
                                SQLError ((SQLHENV) 0, (SQLHDBC) 0, hstmts[h],
(SQLCHAR *) SqlState, &ErrCode, (SQLCHAR *) ErrMsg, 256,
&ErrMsgLen);
                                fprintf (stderr, " %s (%ld) %s\n", SqlState,
ErrCode, ErrMsg);
                            }
                                templong = 0;
                                if
(SQLRowCount (hstmts[h], &templong) != SQL_SUCCESS)
                                    {
                                        fprintf (stderr, " Problem with SQLRowCount\n");
                                        if
(templong != 1)
                                            {
                                                LOG1(1, " %d rows inserted into Lineitem Table in this
insert! \n", totalRows);
                                            }
                                        }
                                }
                            totalRows += templong;
                    }
                }
            }
    }
}

```

```

    }
    lineitemRowsAdded
+= totalRows;
    if (totalRows !=
LinesPerOrder[h][j])
    {
        LOG2(1, "
%d rows inserted into LineItem table in this request, should have
been %d\n", totalRows, LinesPerOrder[h][j]);
    }
    rc =
SQLMoreResults (hstmts[h]);
    if (rc ==
SQL_ERROR || rc == SQL_SUCCESS_WITH_INFO)
    {
        fprintf (stderr, " Problem with SQLMoreResult ins2\n");
        SQLError ((SQLHENV) 0, (SQLHDBC) 0, hstmts[h], (SQLCHAR *)
SqlState, &ErrCode, (SQLCHAR *) ErrMsg, 256, &ErrMsgLen);
        fprintf (stderr, " %s (%ld) %s\n", SqlState, ErrCode,
ErrMsg);
    }
    else if (rc ==
SQL_NO_DATA_FOUND && j < (OrdersInReq[h] - 1))
    {
        fprintf (stderr, " Problem with SQLMoreResults ins3\n");
    }
    OrdersInReq[h] = 0;
}
}
return 0;
}

void
Wait_For_Free_Hstmt (int *h, int inserts)
{
    int i;
    SQLRETURN rc;

    SQLHSTMT newhstmt;
    for (i = 1; i <= maxSess; i++)
        if (!hstmtbusy[i])
            {
                //fprintf(stderr,"Wait_for_Free
returned %d because it wasnt busy\n",i);
                *h = i;
                return;
            }
        rc = SQLWaitForAny (henv, &newhstmt, &i);
        if (rc == SQL_ERROR && (i < 0 || newhstmt ==
SQL_NULL_HSTMT))
            {
                LOG0(5, "Unexpected error: SQLWaitForAny returned
SQL_ERROR in Wait_for_free\n");
                return;
            }
        if (rc == SQL_NO_DATA_FOUND)
            {
                fprintf (stderr, "Bug: SQLWaitForAny
returned complete.\n");
                *h = 1;
                return;
            }
        if (hstmts[i] != newhstmt)
            fprintf (stderr, "SQLWaitForAny returned
garbage\n");
        //fprintf(stderr,"SQLWaitForAny returned %d\n",i);
        if (inserts)
            Get_Insert_Answer (i, rc);
        else
            Get_Delete_Answer (i, rc);
        hstmtbusy[i] = FALSE;
        *h = i;
        return;
}
/*
for (;;)
{
for (i=0;i<maxSess;i++)
{
if (Submit_Req(i)!=SQL_STILL_EXECUTING)

```

```

    {
    *h = i;
    return;
    }
    }
    */
}

SQLRETURN
Submit_Req (int h, int inserts)
{
    SQLRETURN rc;

    if (!hstmtbusy[h])
    {
        SQLSetStmtOption (hstmts[h],
        SQL_ASYNC_ENABLE, SQL_ASYNC_ENABLE_ON);
        hstmtbusy[h] = TRUE;
    }

    rc = SQLExecDirect (hstmts[h], (SQLCHAR *) SQLRequest,
    SQL_NTS);
    if (rc == SQL_STILL_EXECUTING)
    {
        //fprintf(stderr, " Submit_req returns
    SQL_STILL_EXECUTING on %d\n",h);
        hstmtbusy[h] = TRUE;
        return (rc);
    }
    //fprintf(stderr, " Submit_req did not have to wait for
    %d\n",h);

    SQLSetStmtOption (hstmts[h], SQL_ASYNC_ENABLE,
    SQL_ASYNC_ENABLE_OFF);
    hstmtbusy[h] = FALSE;
    if (inserts)
        Get_Insert_Answer (h, rc);
    else
        Get_Delete_Answer (h, rc);
}

```

```

/* From print.c */
/*
 * NOTE: this routine does NOT use the BCD2_* routines. As a
result,
 * it WILL fail if the keys being deleted exceed 32 bits. Since
this
 * would require ~660 update iterations, this seems an acceptable
 * oversight
 */
int
gen_deletes (long min, long cnt, long num, int special)
{
    static int last_num = 0;
    static FILE *dfp = NULL;
    int child = -1;
    DSS_HUGE * sk;
    struct timeb EndTime;
    struct timeb difTime;
    char tempStr[40];
    int rc;
    DSS_HUGE * i;
    DSS_HUGE * temp;

    long o;
    int h;
    long ii;

    BOOL anybusy;

    orderRowsDeleted = 0;
    orderRowsInError = 0;
    lineitemRowsDeleted = 0;
    orderRowsToBeDeleted = 0;

    if (special == 0)
    {
        LOG0(1, "\n Running update function UF2
(deletes)\n");
    }
    else
    {
        LOG0(1, "\n Running special cleanup deletes\n");
    }
    if (flags & VERBOSE)

```

```

        {
            if (special == 0)
                fprintf (stdout, " Running update
function UF2 (deletes)\n");
            else
                fprintf (stdout, " Doing special
cleanup deletes\n");
            //fprintf(stderr," Running update function
UF2 (deletes)\n");
        }
        ftime (&StartTime);
        INIT_HUGE(sk);
        INIT_HUGE(i);
        INIT_HUGE(last);
        INIT_HUGE(start);
        INIT_HUGE(temp);

        if (last_num != num)
        {
            /* Do first time init here */
            last_num = num;
        }

        gen_rng = 1;

        mk_sparse(min, start, special + num / (10000 / refresh));

        HUGE_SUB(start,1,last);

        h = 1;

        DR_Strt ();
        for (child = min; cnt > 0; child++, cnt--)
        {
            mk_sparse(child, sk, special + num / (10000 /
refresh));

            if (gen_rng == 1 && (*sk) - *last == 1 &&
cnt > 1)
                {
                    HUGE_SET(sk,last);

                    continue;
                }
            else
                {
                    if (cnt <= 1 && (*sk) - *last
                    == 1)
                        HUGE_SET(sk,last);

                    SQLRequest[0] = '\0';
                    if (flags & EXPLAIN)
                        strcat (SQLRequest,
                        "EXPLAIN ");
                    //strcat(SQLRequest,"DELETE
FROM ORDERTBL WHERE O_ORDERKEY IN
OrdersInReq[h] = 0;
DR_Strt ();
strcat (SQLRequest, "USING
(");
                    o = 0;
                    for (HUGE_SET(start,i); (*i)
                    <= (*last); (*i)++)
                        {
                            DR_Huge (i);
                            o++;
                            strcat
                            sprintf (tempStr,
                            "INT");
                            if (*i != *last)
                                strcat
                                if (cnt <= 1 && (*sk) > *last)
                                    {
                                        strcat
                                        DR_Huge (sk);
                                        o++;
                                        strcat
                                        sprintf (tempStr,
                                        "O");
                                        strcat
                                    }
                                }
                    }
                }

```

```

            strcat
(SQLRequest, " INT");
    }
    strcat (SQLRequest, " ");
    o = 0;
    for (HUGE_SET(start,i); (*i)
<= *last; (*i)++)
        {
            o++;

            orderRowsToBeDeleted++;

            OrdersInReq[h]++;
            strcat
(SQLRequest, "DELETE FROM ORDERTBL WHERE O_ORDERKEY =:O");
            sprintf (tempStr,
"%ld", o);
            strcat
(SQLRequest, tempStr);
            //if (i!=last)
            strcat
(SQLRequest, ";");

            }
            if (cnt <= 1 && (*sk) > *last)
            {
                o++;

                orderRowsToBeDeleted++;

                OrdersInReq[h]++;
                strcat
(SQLRequest, "DELETE FROM ORDERTBL WHERE O_ORDERKEY =:O");
                sprintf (tempStr,
"%ld", o);
                strcat
(SQLRequest, tempStr);
                //if (i!=last)
                strcat
(SQLRequest, ";");

                }
                //strcat(SQLRequest, ";");
                o = 0;
                for (HUGE_SET(start,i); (*i)
<= *last; (*i)++)
                    {
                        strcat
(SQLRequest, "DELETE FROM LINEITEM WHERE L_ORDERKEY =:O");
                        sprintf (tempStr,
"%ld", o);
                        strcat
(SQLRequest, tempStr);
                        //if (i!=last)
                        strcat
(SQLRequest, ";");

                        }
                        if (cnt <= 1 && (*sk) > *last)
                        {
                            o++;

                            strcat
(SQLRequest, "DELETE FROM LINEITEM WHERE L_ORDERKEY =:O");
                            sprintf (tempStr,
"%ld", o);
                            strcat
(SQLRequest, tempStr);
                            //if (i!=last)
                            strcat
(SQLRequest, ";");

                            }
                            LOGl(1, "SQL
COMMAND:\n\t%s\n", SQLRequest);
                            DR_End (LINE);

                            Submit_Req (h, FALSE);
                            Wait_For_Free_Hstmt (&hstmt,
FALSE);
                            hstmt = hstmts[h];
                            DR_Strt ();
                            //fprintf(stderr, " Next req
goes on hstmt %d\n",h);

                            HUGE_SET(sk,start);
                            HUGE_SET(sk,last);
                        }
                    }
                    //fprintf(stderr, " All submitted, waiting\n");
                    anybusy = TRUE;
                    while (anybusy)

```

```

    {
        anybusy = FALSE;
        for (h = 1; h <= maxSess; h++)
            {
                if (hstmtbusy[h])
                    {
                        rc =
SQLWaitForAny (henv, &hstmt, (int *) &ii);
                        if (rc ==
SQL_ERROR)
                            {
                                fprintf (stderr, "SQLWaitForAny returned SQL_ERROR\n");
                                break;
                            }
                        if (rc ==
SQL_NO_DATA_FOUND)
                            break;
                        if (hstmts[ii] !=
hstmt)
                            fprintf
(stderr, "SQLWaitForAny returned garbage\n");

                        //fprintf(stderr, "SQLWaitForAny returned %d\n", ii);
                        Get_Delete_Answer
(ii, (SQLRETURN) rc);
                    }
                if (hstmtbusy[h])
                    anybusy = TRUE;
            }
        }

        hstmt = hstmts[1];
        ftime (&EndTime);
        difTime.time = EndTime.time - StartTime.time;
        if (EndTime.millitm >= StartTime.millitm)
            {
                difTime.millitm = (unsigned short)
(EndTime.millitm - StartTime.millitm);
            }
        else
            {

```

```

                difTime.millitm = (unsigned short) ((1000 +
EndTime.millitm) - StartTime.millitm);
                difTime.time -= 1;
            };
        UF2time = difTime.time + difTime.millitm / 1000.0 + 0.005;

        StartTime.time = EndTime.time;
        StartTime.millitm = EndTime.millitm;

        if (orderRowsDeleted == 0 && special)
            UF2time = 0.0;
        else
            {
                LOG2(1, " UF2 Time was %.2f seconds. Ended
at %s\n\n", UF2time, ctime (&StartTime.time));

                LOG1(1, " Total %d deleted from Order
table\n", orderRowsDeleted);
                if (orderRowsDeleted !=
orderRowsToBeDeleted)
                    {
                        LOG1(1, " Should have been %d
deleted from Order table!!\n", orderRowsToBeDeleted);
                    }
                LOG1(1, " Total %d deleted from LineItem
table\n", lineitemRowsDeleted);
                if (orderRowsInError > 0)
                    {
                        LOG1(1, " Total %d deletes failed
due to database errors\n", orderRowsInError);
                    }
            }

        return (0);
    }

    int
gen_deletes_via_fastload (long min, long cnt, long num, int
special)
    {
        static int last_num = 0;

```

```

static FILE *dfp = NULL;
int child = -1;

char SqlState[6];
char ErrMsg[256];
short ErrMsgLen;
long ErrCode;
BOOL retry;

struct timeb EndTime;
struct timeb difTime;
DSS_HUGE * sk;

int h;

INIT_HUGE(sk);
orderRowsDeleted = 0;
lineitemRowsDeleted = 0;
orderRowsToBeDeleted = 0;

prep_direct((db_name)?db_name:DBNAME);

if (special == 0)
{
LOGO(1, "\n Running update function UF2
(deletes)\n");
}
else
{
LOGO(1, "\n Running special cleanup deletes\n");
}
if (flags & VERBOSE)
{
if (special == 0)
fprintf (stdout, " Running update
function UF2 (deletes)\n");
else
fprintf (stdout, " Doing special
cleanup deletes\n");
//fprintf(stderr," Running update function
UF2 (deletes)\n");
}
ftime (&StartTime);

```

```

/* In this version of UF2, we first use the Teradata
"fastload" utility
to send the keys to be deleted to a temporary table on
the Teradata system.
After all the rows have been sent, we use two DELETE
statements
to delete the rows from the existing tables */

if (last_num != num)
{
/* Do first time init here */
last_num = num;
}
gen_rng = 1;
INIT_HUGE(start);

mk_sparse(min, start , special + num / (10000 / refresh));
last = start - 1;
h = 1;

DR_Strt ();
for (child = min; cnt > 0; child++, cnt--)
{
mk_sparse (child, sk, special + num / (10000 /
refresh));

ld_deletes (sk, upd_num);
orderRowsToBeDeleted++;
}

close_direct();

hstmt = hstmts[1];

/* By doing both DELETE statements in a single request to
the Teradata system,
Teradata guarantees these are done as an atomic
transaction */

retry = TRUE;

```

```

while (retry)
{
    retry = FALSE;
    if (SQLExecDirect (hstmt, (unsigned char *)
#ifdef DOL
        "DELETE FROM ORDERTBL WHERE O_ORDERKEY =
KEYS_TO_DELETE.ORDERKEY;DELETE FROM LINEITEM WHERE L_ORDERKEY =
KEYS_TO_DELETE.ORDERKEY",
#else
        "DELETE FROM LINEITEM WHERE L_ORDERKEY =
KEYS_TO_DELETE.ORDERKEY;DELETE FROM ORDERTBL WHERE O_ORDERKEY =
KEYS_TO_DELETE.ORDERKEY",
#endif
        SQL_NTS) == SQL_ERROR)
    {
        LOG0(3, " UF2 delete failed\n");
        SQLError ((SQLHENV) 0, (SQLHDBC) 0,
hstmt, (SQLCHAR *) SqlState, &ErrCode, (SQLCHAR *) ErrMsg, 256,
&ErrMsgLen);
        LOG2(3, " %d: %s\n", ErrCode,
ErrMsg);
        SQLFreeStmt (hstmt, SQL_CLOSE);
        if (canRetry(ErrCode)) /*
Transaction ABORTed due to DeadLock or crash */
        {
            retry = TRUE;
            LOG0(5, " Attempting to
resubmit UF2 delete\n");
        }
    }
    else
    {
#ifdef DOL
        if (SQLRowCount (hstmt,
&orderRowsDeleted) != SQL_SUCCESS)
            fprintf (stderr, " SQLRowCount
problem\n");
#else
        if (SQLRowCount (hstmt,
&lineitemRowsDeleted) != SQL_SUCCESS)
            fprintf (stderr, " SQLRowCount
problem\n");
#endif
        if (SQLMoreResults (hstmt) !=
SQL_SUCCESS)

```

```

            fprintf (stderr, "
SQLMoreResults problem\n");
#ifdef DOL
        if (SQLRowCount (hstmt,
&lineitemRowsDeleted) != SQL_SUCCESS)
            fprintf (stderr, " SQLRowCount
problem\n");
#else
        if (SQLRowCount (hstmt,
&orderRowsDeleted) != SQL_SUCCESS)
            fprintf (stderr, " SQLRowCount
problem\n");
#endif
    }
}

ftime (&EndTime);
difTime.time = EndTime.time - StartTime.time;
if (EndTime.millitm >= StartTime.millitm)
{
    difTime.millitm = (unsigned short)
(EndTime.millitm - StartTime.millitm);
}
else
{
    difTime.millitm = (unsigned short) ((1000 +
EndTime.millitm) - StartTime.millitm);
    difTime.time -= 1;
};
UF2time = difTime.time + difTime.millitm / 1000.0 + 0.005;

StartTime.time = EndTime.time;
StartTime.millitm = EndTime.millitm;

if (orderRowsDeleted == 0 && special)
    UF2time = 0.0;
else
{
    LOG2(1, " UF2 Time was %.2f seconds. Ended
at %s\n\n", UF2time, ctime (&StartTime.time));

    LOG1(1, " Total %d deleted from Order
table\n", orderRowsDeleted);
    if (orderRowsDeleted !=
orderRowsToBeDeleted)

```



```

        {
            LOG1(1, " Should have been %d
deleted from Order table!!\n", orderRowsToBeDeleted);
        }
        LOG1(1, " Total %d deleted from LineItem
table\n", lineitemRowsDeleted);
    }

    return (0);
}

int
Build_Multistatement_Insert (int h)
{
    char tstr2[6] = "?";
    char tstr1[3] = "?";
    char tstr3[4] = "?";
    int j;
    int k;

    /*if ((flags & VERBOSE)&&(specialtest))
    {
        LOG2(1, Building multi insert, orders %d, lines[0]
=%d\n",OrdersInReq,LinesPerOrder[0]);
    }
    */
    SQLRequest[0] = '\0';
    if (flags & EXPLAIN)
        strcpy (SQLRequest, " EXPLAIN ");
    sprintf (tstr2, "%d", LinesPerOrder[h][1] + 1);
    strcat (SQLRequest, "/*");
    strcat (SQLRequest, tstr2);
    strcat (SQLRequest, "*/ ");
    strcat (SQLRequest, "USING ");

    if (OrdersInReq[h] > 10 || OrdersInReq[h] > setsPerReq ||
OrdersInReq[h] <= 0)
    {
        fprintf (stderr, "OrdersInReq[h] = %d\n",
OrdersInReq[h]);
        fclose (fdetail);
    }
    }

    fclose (fsummary);
    exit (1);
}

// First, build the using info for all the ORDER inserts
for (k = 0; k < OrdersInReq[h]; k++)
{
    tstr1[0] = (char) ('0' + k);
    tstr1[1] = '\0';
    if (k > 0)
        strcat (SQLRequest, ",");
    strcat (SQLRequest, "OOK");
    strcat (SQLRequest, tstr1);
    if (scale > 300)
        strcat (SQLRequest, "(DEC(15,0))");
    else
        strcat (SQLRequest, "(INT)");
    strcat (SQLRequest, ",OCK");
    strcat (SQLRequest, tstr1);
    strcat (SQLRequest, "(INT)");
    strcat (SQLRequest, ",OOS");
    strcat (SQLRequest, tstr1);
    strcat (SQLRequest, "(CHAR(1))");
    strcat (SQLRequest, ",OTP");
    strcat (SQLRequest, tstr1);
    strcat (SQLRequest, "(DEC(15,2))");
    strcat (SQLRequest, ",OOD");
    strcat (SQLRequest, tstr1);
    strcat (SQLRequest, "(DATE)");
    strcat (SQLRequest, ",OOP");
    strcat (SQLRequest, tstr1);
    strcat (SQLRequest, "(CHAR(15))");
    strcat (SQLRequest, ",OC");
    strcat (SQLRequest, tstr1);
    strcat (SQLRequest, "(CHAR(15))");
    strcat (SQLRequest, ",OSP");
    strcat (SQLRequest, tstr1);
    strcat (SQLRequest, "(INT)");
    strcat (SQLRequest, ",OCM");
    strcat (SQLRequest, tstr1);
    strcat (SQLRequest, "(VARCHAR(79)) ");
    if (LinesPerOrder[h][k] > 20)
    {
        fprintf (stderr, "
LinesPerOrder[h][k] > 20, %d\n", LinesPerOrder[h][k]);
        fclose (fdetail);
        fclose (fsummary);
    }
}

```

```

        exit (1);
    }
}

// Now build all the using data for the lineitems.
for (j = 0; j < 20; j++)
    for (k = 0; k < OrdersInReq[h]; k++)
    {
        if (j < LinesPerOrder[h][k])
        {
            tstr1[0] = (char) ('0' + k);
            tstr1[1] = '\0';
            tstr3[0] = (char) ('0' + j /
10);
            tstr3[1] = (char) ('0' + j %
10);
            tstr3[2] = '\0';
            strcpy (tstr2, tstr1);
            strcat (tstr2, tstr3);

            strcat (SQLRequest, ",LOK");
            strcat (SQLRequest, tstr2);
            if (scale > 300)
                strcat (SQLRequest,
"(DEC(15,0))");
            else
                strcat (SQLRequest,
"(INT)");

            strcat (SQLRequest, ",LPK");
            strcat (SQLRequest, tstr2);
            strcat (SQLRequest, "(INT)");

            strcat (SQLRequest, ",LSK");
            strcat (SQLRequest, tstr2);
            strcat (SQLRequest, "(INT)");
            strcat (SQLRequest, ",LLN");
            strcat (SQLRequest, tstr2);
            strcat (SQLRequest, "(INT)");
            strcat (SQLRequest, ",LQ");
            strcat (SQLRequest, tstr2);
            strcat (SQLRequest, "(INT)");
            strcat (SQLRequest, ",LEP");
            strcat (SQLRequest, tstr2);

            strcat (SQLRequest,
"(DEC(15,2))");
            strcat (SQLRequest, ",LDSC");
            strcat (SQLRequest, tstr2);
            strcat (SQLRequest,
"(DEC(15,2))");
            strcat (SQLRequest, ",LTAX");
            strcat (SQLRequest, tstr2);
            strcat (SQLRequest,
"(DEC(15,2))");
            strcat (SQLRequest, ",LRTNF");
            strcat (SQLRequest, tstr2);
            strcat (SQLRequest,
"(CHAR(1))");
            strcat (SQLRequest, ",LLS");
            strcat (SQLRequest, tstr2);
            strcat (SQLRequest,
"(CHAR(1))");
            strcat (SQLRequest, ",LSHD");
            strcat (SQLRequest, tstr2);
            strcat (SQLRequest, "(DATE)");
            strcat (SQLRequest, ",LCMTD");
            strcat (SQLRequest, tstr2);
            strcat (SQLRequest, "(DATE)");
            strcat (SQLRequest, ",LRCD");
            strcat (SQLRequest, tstr2);
            strcat (SQLRequest, "(DATE)");
            strcat (SQLRequest, ",LSHI");
            strcat (SQLRequest, tstr2);
            strcat (SQLRequest,
"(CHAR(25))");
            strcat (SQLRequest, ",LSHM");
            strcat (SQLRequest, tstr2);
            strcat (SQLRequest,
"(CHAR(10))");
            strcat (SQLRequest, ",LCM");
            strcat (SQLRequest, tstr2);
            strcat (SQLRequest,
"(VARCHAR(44)) ");
        }
    }

// Next, build the inserts for all the ORDER inserts.
for (k = 0; k < OrdersInReq[h]; k++)
    {
        tstr1[0] = (char) ('0' + k);

```

```

VALUES(");
    tstr1[1] = '\0';
    strcat (SQLRequest, " INSERT INTO ORDERTBL

    strcat (SQLRequest, ":OOK");
    strcat (SQLRequest, tstr1);
    strcat (SQLRequest, ",:OCK");
    strcat (SQLRequest, tstr1);
    strcat (SQLRequest, ",:OOS");
    strcat (SQLRequest, tstr1);
    strcat (SQLRequest, ",:OTP");
    strcat (SQLRequest, tstr1);
    strcat (SQLRequest, ",:OOD");
    strcat (SQLRequest, tstr1);
    strcat (SQLRequest, ",:OOP");
    strcat (SQLRequest, tstr1);
    strcat (SQLRequest, ",:OC");
    strcat (SQLRequest, tstr1);
    strcat (SQLRequest, ",:OSP");
    strcat (SQLRequest, tstr1);
    strcat (SQLRequest, ",:OCM");
    strcat (SQLRequest, tstr1);
    strcat (SQLRequest, ");");
}
for (j = 0; j < 20; j++)
    for (k = 0; k < OrdersInReq[h]; k++)
    {
        if (j < LinesPerOrder[h][k])
        {
            tstr1[0] = (char) ('0' + k);
            tstr1[1] = '\0';
            tstr3[0] = (char) ('0' + j /
            tstr3[1] = (char) ('0' + j %
            tstr3[2] = '\0';
            strcpy (tstr2, tstr1);
            strcat (tstr2, tstr3);

            strcat (SQLRequest, " INSERT
            strcat (SQLRequest, ":LOK");
            strcat (SQLRequest, tstr2);
            strcat (SQLRequest, ",:LPK");
            strcat (SQLRequest, tstr2);
            strcat (SQLRequest, ",:LSK");
            strcat (SQLRequest, tstr2);
            strcat (SQLRequest, ",:LLN");

            strcat (SQLRequest, tstr2);
            strcat (SQLRequest, ",:LQ");
            strcat (SQLRequest, tstr2);
            strcat (SQLRequest, ",:LEP");
            strcat (SQLRequest, tstr2);
            strcat (SQLRequest, ",:LDSC");
            strcat (SQLRequest, tstr2);
            strcat (SQLRequest, ",:LTAX");
            strcat (SQLRequest, tstr2);
            strcat (SQLRequest,
            strcat (SQLRequest, tstr2);
            strcat (SQLRequest, ",:LLS");
            strcat (SQLRequest, tstr2);
            strcat (SQLRequest, ",:LSHD");
            strcat (SQLRequest, tstr2);
            strcat (SQLRequest,
            strcat (SQLRequest, tstr2);
            strcat (SQLRequest, ",:LRCD");
            strcat (SQLRequest, tstr2);
            strcat (SQLRequest, ",:LSHI");
            strcat (SQLRequest, tstr2);
            strcat (SQLRequest, ",:LSHM");
            strcat (SQLRequest, tstr2);
            strcat (SQLRequest, ",:LCM");
            strcat (SQLRequest, tstr2);
            strcat (SQLRequest, ");");
        }
    }
return (0);
}

/*
 * generate a particular table
 */
void
gen_updates (long start, long count, long upd_num)
{
    long i;

    int h;

```

```

static int completed = 0;

struct timeb EndTime;
struct timeb difTime;
int rc;
long maxReqLen = 0;
long maxUsingLen = 0;

BOOL anybusy;
DSS_HUGE * sk;
int j;
int k;

orderRowsAdded = 0;
orderRowsInError = 0;
lineitemRowsAdded = 0;
orderRowsToBeAdded = 0;
lineitemRowsToBeAdded = 0;
duplicates = 0;

LOGO(1, "\n Running update function UF1 (inserts)\n");
if (flags & VERBOSE)
{
    fprintf (stdout, " Running update function
UF1 (inserts)\n");
    //fprintf(stderr," Running update function
UF1 (inserts)\n");
}
ftime (&StartTime);

INIT_HUGE(sk);

for (i = 1; i <= maxSess; i++)
{
    hstmtbusy[i] = FALSE;
    OrdersInReq[i] = 0;
    LinesPerOrder[i][0] = 0;
    LinesPerOrder[i][1] = 0;
    for (k=0;k<20;k++)
    {
        INIT_HUGE(o[i][k].okey);
        for (j=0; j < O_LCNT_MAX; j++)
            INIT_HUGE(o[i][k].l[j].okey);
    }
    h = 1;
    hstmt = hstmts[1];

    DR_Strt ();
    for (i = start; count; count--, i++)
    {
        LIFENOISE (1000, i);

        mk_sparse (i, sk, 1 + upd_num / (10000 /
refresh));

        mk_order (sk, &o[h][OrdersInReq[h]]);

        LinesPerOrder[h][OrdersInReq[h]] =
o[h][OrdersInReq[h]].lines;
        orderRowsToBeAdded++;
        lineitemRowsToBeAdded +=
o[h][OrdersInReq[h]].lines;

        OrdersInReq[h]++;
        LinesPerOrder[h][OrdersInReq[h]] = 0;

        if ((OrdersInReq[h] >= setsPerReq) || (count
<= 1))
        {
            int k;
            int j;
            Build_Multistatement_Insert
(h);
            for (k = 0; k <
OrdersInReq[h]; k++)
                xld_order (&o[h][k],
upd_num);
        }
    }
}

```

```

                for (j = 0; j < 20; j++)
                    for (k = 0; k <
OrdersInReq[h]; k++)
                    if (j <
LinesPerOrder[h][k])
                        xld_one_line (&o[h][k], j, upd_num);
                if (strlen (SQLRequest) >
(size_t) maxReqLen)
                    maxReqLen = strlen
(SQLRequest);
                if (((int) (p1 - UsingBuffer))
> maxUsingLen)
                    maxUsingLen = (int) (p1
- UsingBuffer);
                Submit_Req (h, TRUE);
                Wait_For_Free_Hstmt (&h,
TRUE);
                hstmt = hstmts[h];
                DR_Strt ();
                OrdersInReq[h] = 0;
                //fprintf(stderr," Next req
goes on hstmt %d\n",h);
            }
        }
        //fprintf(stderr," All submitted, waiting\n");
        anybusy = TRUE;
        while (anybusy)
        {
            anybusy = FALSE;
            for (h = 1; h <= maxSess; h++)
            {
                if (hstmtbusy[h])
                {
                    rc =
                    if (rc ==
SQL_WAIT_FOR_ANY_ERROR)
                    {
                        fprintf (stderr, "SQLWaitForAny returned SQL_ERROR\n");
                        break;
                    }
                }
            }
        }
        if (rc ==
SQL_NO_DATA_FOUND)
            break;
        if (hstmts[i] !=
hstmt)
            fprintf
(stderr, "SQLWaitForAny returned garbage\n");
        //fprintf(stderr,"SQLWaitForAny returned %d\n",i);
        Get_Insert_Answer
(i, (SQLRETURN) rc);
    }
    if (hstmtbusy[h])
        anybusy = TRUE;
}
hstmt = hstmts[1];
ftime (&EndTime);
difTime.time = EndTime.time - StartTime.time;
if (EndTime.millitm >= StartTime.millitm)
{
    difTime.millitm = (unsigned short)
(EndTime.millitm - StartTime.millitm);
}
else
{
    difTime.millitm = (unsigned short) ((1000 +
EndTime.millitm) - StartTime.millitm);
    difTime.time -= 1;
};
UF1time = difTime.time + difTime.millitm / 1000.0 + 0.005;
StartTime.time = EndTime.time;
StartTime.millitm = EndTime.millitm;
LOG2(1, " UF1 Insert Time was %.2f seconds. Ended at
%s\n\n", UF1time, ctime (&StartTime.time));
#ifdef _DEBUG
if (flags & VERBOSE)

```

```

        fprintf (stdout, " Insert stmt max len %d, using
len %d\n", maxReqLen, maxUsingLen);
#endif

    LOGl(1, " Total %d inserted into Order table\n",
orderRowsAdded);
    if (orderRowsAdded != orderRowsToBeAdded)
        {
            LOGl(1, " Should have been %d inserted into Order
table!!\n", orderRowsToBeAdded);
        }
    LOGl(1, " Total %d inserted into LineItem table\n",
lineitemRowsAdded);
    if (lineitemRowsAdded != lineitemRowsToBeAdded)
        {
            LOGl(1, " Should have been %d inserted into
LineItem table\n", lineitemRowsToBeAdded);
        }

    if ((flags & VERBOSE) || (duplicates > 0))
        {
            LOGl(1, " Total %d inserts ignored due to
duplicate unique primary index\n", duplicates);
        }

    if (orderRowsInError > 0)
        {
            LOGl(1, " Total %d inserts failed due to database
errors\n", orderRowsInError);
        }
}

/*
 * generate a particular table
 */
void
gen_updates_via_fastload (long start, long count, long upd_num)
{
    long i;

    static int completed = 0;
    order_t o;

    char SqlState[6];
    char ErrMsg[256];
    short ErrMsgLen;

```

```

    long ErrCode;
    BOOL retry;

    struct timeb EndTime;
    struct timeb difTime;
    DSS_HUGE *sk;

    orderRowsAdded = 0;
    lineitemRowsAdded = 0;
    orderRowsToBeAdded = 0;
    lineitemRowsToBeAdded = 0;

    prep_direct((db_name)?db_name:DBNAME);

    LOG0(0, "\n Running update function UF1 (inserts)\n");
    if (flags & VERBOSE)
        {
            fprintf (stdout, " Running update function
UF1 (inserts)\n");
            //fprintf(stderr, " Running update function
UF1 (inserts)\n");
        }
    ftime (&StartTime);

    /* In this version of UF1, we first use the Teradata
"fastload" utility
    to send the rows to be inserted to temporary tables on
the Teradata system.
    After all the rows have been sent, we use INSERT/SELECT
statements
    to merge the new rows with the existing tables */

    INIT_HUGE(o.okey);
    INIT_HUGE(sk);
    for (i=0; i < O_LCNT_MAX; i++)
        INIT_HUGE(o.l[i].okey);

    for (i = start; count; count--, i++)
        {
            LIFENOISE (1000, i);

```

```

refresh));
        mk_sparse (i, sk, 1 + upd_num / (10000 /

mk_order (sk, &o);

orderRowsToBeAdded++;
lineitemRowsToBeAdded += o.lines;

ld_order_line (&o, upd_num);

}

close_direct();

hstmt = hstmts[1];

/* By doing both INSERT/SELECT statements in a single
request to the Teradata system,
Teradata guarantees these are done as an atomic
transaction */

retry = TRUE;
while (retry)
{
    retry = FALSE;
    if (SQLExecDirect (hstmt, (unsigned char *)
#ifdef IOL
        "INSERT INTO ORDERTBL SELECT * FROM
ORDERTBL_UPDATES;INSERT INTO LINEITEM SELECT * FROM
LINEITEM_UPDATES",
#else
        "INSERT INTO LINEITEM SELECT * FROM
LINEITEM_UPDATES;INSERT INTO ORDERTBL SELECT * FROM
ORDERTBL_UPDATES",
#endif
        SQL_NTS) == SQL_ERROR)
    {
        LOG0(3, " Uf1 Insert/select
failed\n");

```

```

        SQLError ((SQLHENV) 0, (SQLHDBC) 0,
hstmt, (SQLCHAR *) SqlState, &ErrCode, (SQLCHAR *) ErrMsg, 256,
&ErrMsgLen);
        LOG2(3, " %d: %s\n", ErrCode,
ErrMsg);
        SQLFreeStmt (hstmt, SQL_CLOSE);
        if (canRetry(ErrCode)) /*
Transaction ABORTed due to DeadLock or crash */
        {
            retry = TRUE;
            LOG0(3, " Attempting to
resubmit Uf1 Insert/select\n");
        }
    else
    {
#ifdef IOL
        if (SQLRowCount (hstmt,
&orderRowsAdded) != SQL_SUCCESS)
            fprintf (stderr, " SQLRowCount
problem\n");
#else
        if (SQLRowCount (hstmt,
&lineitemRowsAdded) != SQL_SUCCESS)
            fprintf (stderr, "
SQLRowCount problem\n");
#endif

        if (SQLMoreResults (hstmt) !=
SQL_SUCCESS)
            fprintf (stderr, "
SQLMoreResults problem\n");
#ifdef IOL
        if (SQLRowCount (hstmt,
&lineitemRowsAdded) != SQL_SUCCESS)
            fprintf (stderr, " SQLRowCount
problem\n");
#else
        if (SQLRowCount (hstmt,
&orderRowsAdded) != SQL_SUCCESS)
            fprintf (stderr, "
SQLRowCount problem\n");
#endif
    }
}

```

```

ftime (&EndTime);
difTime.time = EndTime.time - StartTime.time;
if (EndTime.millitm >= StartTime.millitm)
    {
        difTime.millitm = (unsigned short)
(EndTime.millitm - StartTime.millitm);
    }
else
    {
        difTime.millitm = (unsigned short) ((1000 +
EndTime.millitm) - StartTime.millitm);
        difTime.time -= 1;
    };
UF1time = difTime.time + difTime.millitm / 1000.0 + 0.005;

StartTime.time = EndTime.time;
StartTime.millitm = EndTime.millitm;

LOG2(1, " UF1 Insert Time was %.2f seconds. Ended at
%s\n\n", UF1time, ctime (&StartTime.time));

LOG1(1, " Total %d inserted into Order table\n",
orderRowsAdded);
if (orderRowsAdded != orderRowsToBeAdded)
    {
        LOG1(1, " Should have been %d inserted into Order
table!!\n", orderRowsToBeAdded);
    }
LOG1(1, " Total %d inserted into LineItem table\n",
lineitemRowsAdded);
if (lineitemRowsAdded != lineitemRowsToBeAdded)
    {
        LOG1(1, " Should have been %d inserted into
LineItem table\n", lineitemRowsToBeAdded);
    }
}

void
stop_proc (int signum)
{
    fflush (fdetail);
    fflush (fsummary);

    fflush (stdout);
    fprintf (stderr, " Program terminating abnormally!
signal=%d\n", signum);
    fflush (stderr);
    exit (1);
}

int
Connect_Sessions ()
{
    char SqlState[6];
    char ErrMsg[256];
    short ErrMsgLen;
    long ErrCode;

    int i;
    char szLogon[240];
    SQLCHAR szDSN[32];
    SQLCHAR szUID[32];
    SQLCHAR szPWD[32];
    char tempstr[100];

    char *place;
    char *place2;

    szDSN[0] = '\0';
    szUID[0] = '\0';
    szPWD[0] = '\0';
    szLogon[0] = '\0';

    strcpy (szLogon, db_name);

    if (strstr (szLogon, "/") != NULL)
        {
            place = strstr (szLogon, "/");
            place[0] = '\0';
            strcpy ((char *) szDSN, szLogon);
            place++;
            place2 = strstr (place, ",");
            place2[0] = '\0';
            strcpy ((char *) szUID, place);
            place2++;
            strcpy ((char *) szPWD, place2);
        }
}

```



```

    }
    if (SQLAllocEnv (&henv) != SQL_SUCCESS)
    {
        fprintf (stderr, "Could not allocate
ENV\n");
        exit (1);
    }

    for (i = 1; i <= maxSess; i++)
    {
        if (SQLAllocConnect (henv, &hdbc) !=
SQL_SUCCESS)
            {
                fprintf (stderr, "Could not
allocate DBC\n");
                exit (1);
            }
        if (SQLConnect (hdbc, szDSN, SQL_NTS, szUID,
SQL_NTS, szPWD, SQL_NTS) != SQL_SUCCESS)
            {
                if (i > 1)
                    fprintf (stderr, " When
trying to log on the %dth session, \n", i);
                fprintf (stderr, "Could not
log on using '%s/%s,%s'\n", szDSN, szUID, szPWD);
                SQLError ((SQLHENV) 0, hdbc,
(SQLHSTMT) 0, (SQLCHAR *) SqlState, &ErrCode, (SQLCHAR *) ErrMsg,
256, &ErrMsgLen);
                fprintf (stderr, " %s (%ld)
%s\n", SqlState, ErrCode, ErrMsg);
                if (i < 10)
                    exit (1);
                fprintf (stderr, " Continuing
with %d sessions only\n", i - 1);
                fflush (stderr);
                maxSess = i - 1;
                hdbc = hdbcs[1];
                hstmt = hstmts[1];

                return (0);
            }
        if (SQLAllocStmt (hdbc, &hstmt) !=
SQL_SUCCESS)
            {

```

```

                fprintf (stderr, "Could not
allocate STMT\n");
                exit (1);
            }
        hdbcs[i] = hdbc;
        hstmts[i] = hstmt;
        if (updates == 0)
            break;
    }
    hdbc = hdbcs[1];
    hstmt = hstmts[1];

    return (0);
}
int
Disconnect_Sessions ()
{
    int i;
    for (i = 1; i <= maxSess; i++)
        {
            SQLFreeStmt (hstmts[i], SQL_DROP);
            hstmt = NULL;
            if (SQLDisconnect (hdbcs[i]) == SQL_ERROR)
                fprintf (stderr, "Error when logging
off the sessions\n");
            if (updates == 0)
                break;
        }

    if (flags & VERBOSE)
        fprintf (stdout, "Sessions are now logged off from
Teradata\n");
    return (0);
}
int
Run_Query (const char *SQLReq, int qnum)
{
    static char SQLR[2048];
    char SqlState[6];
    char ErrMsg[256];
    short ErrMsgLen;
    long ErrCode;
    BOOL retry;

    struct timeb EndTime;

```

```

struct timeb difTime;
int i;
char *SQ1;
char *SQ2;
const char *SQ;
strcpy (SQLR, SQLReq);

fprintf (fdetail, "\n Submitting SQL Request #d:\n\n",
qnum);
fprintf (stdout, " Submitting SQL Request #d:\n", qnum);

SQ2 = SQLR;

while (SQ2 != NULL)
{
    SQ1 = SQ2;
    for (i = strlen (SQ1) - 1; i > 0; i--) /*
Trim trailing spaces */
        if (SQ1[i] == ' ' ||
            SQ1[i] == '\r' ||
            SQ1[i] == '\t')
            SQ1[i] = '\0';
        else
            break;
    if (SQ1[strlen (SQ1) - 1] == ';') /* Trim
trailing semicolon and spaces */
    {
        SQ1[strlen (SQ1) - 1] = '\0';
        for (i = strlen (SQ1) - 1; i >
0; i--)
            if (SQ1[i] == ' ' ||
                SQ1[i] == '\r' ||
                SQ1[i] == '\t')
                SQ1[i] = '\0';
            else
                break;
    }

    if (strstr (SQ1, "/*") != NULL)
        SQ2 = strstr (strstr (strstr (SQ1,
"/**"), "**/"), ";"); /* look past the comment strings! */
    else
        SQ2 = strstr (SQ1, ";"); /* If
semicolon, this is a multi-statement request */
    if (SQ2 != NULL)
        {

```

```

        *SQ2 = '\0'; /* Break out the
first SQL statement in SQ1 */
        SQ2++; /* SQ2 points to
the next statement */
        }
        SQ = SQ1;
        while (SQ[0] != 0)
        {
            if (SQ[0] == '\r')
                fputc ('\n', fdetail);
            else
                fputc (SQ[0], fdetail);
            SQ++;
        }
        fprintf (fdetail, ";\n\n");

        retry = TRUE;
        while (retry)
        {
            retry = FALSE;
            if (SQLExecDirect (hstmt, (unsigned
char *) SQ1, SQL_NTS) == SQL_ERROR)
            {
                fprintf (fdetail, "
Query #d failed\n", qnum);
                SQLError ((SQLHENV) 0,
(SQLHDBC) 0, hstmt, (SQLCHAR *) SqlState, &ErrCode, (SQLCHAR *)
ErrMsg, 256, &ErrMsgLen);
                fprintf (fdetail, " %d:
%s\n", ErrCode, ErrMsg);
                fprintf (stdout, " Query
#d failed\n", qnum);
                fprintf (stdout, " %d:
%s\n", ErrCode, ErrMsg);
                SQLFreeStmt (hstmt,
SQL_CLOSE);
                if (canRetry(ErrCode))
                    /* Transaction ABORTed due to DeadLock or crash */
                    {
                        retry =
TRUE;
                        fprintf
(fdetail, " Attempting to resubmit Query #d\n", qnum);
                        fprintf
(stdout, " Attempting to resubmit Query #d\n", qnum);

```

```

        }
        else
            if (ErrCode != 3803) /*
table already exists */
                {
                    QueryTime[qnum] = 0.0;
                }
            return (-
1);
        }
    else
    {
        Fetch_and_Display_All_Rows (hstmt, qnum);
    }
}

ftime (&EndTime);
difTime.time = EndTime.time - StartTime.time;
if (EndTime.millitm >= StartTime.millitm)
{
    difTime.millitm = (unsigned short)
(EndTime.millitm - StartTime.millitm);
}
else
{
    difTime.millitm = (unsigned short) ((1000 +
EndTime.millitm) - StartTime.millitm);
    difTime.time -= 1;
};

QueryTime[qnum] = difTime.time + difTime.millitm / 1000.0
+ 0.005;

StartTime.time = EndTime.time;
StartTime.millitm = EndTime.millitm;

fprintf (fdetail, "Time was %.2f seconds. Query ended at
%s\n\n", QueryTime[qnum], ctime (&StartTime.time));

return (0);
}

```

```

int
Verify_Database ()
{
    long actualLen;
    long crowcount;
    double newsF;
    char SqlState[6];
    char ErrMsg[256];
    char InfoData[256];
    short ErrMsgLen;
    long ErrCode;
    long vAMPs;
    int isV2;

    isV2 = 0;
    if (SQLExecDirect (hstmt, (SQLCHAR *) "SELECT INFODATA
FROM DBC.DBCINFO WHERE INFOKEY='RELEASE'", SQL_NTS) != SQL_ERROR)
    {
        if (SQLFetch (hstmt) == SQL_SUCCESS)
            if (SQLGetData (hstmt, 1, SQL_C_CHAR,
InfoData, 256, &actualLen) == SQL_SUCCESS)
            {
                fprintf (fdetail, " The
system under test is running Teradata release %s\n", InfoData);
                fprintf (stdout, " The
system under test is running Teradata release %s\n", InfoData);
                if (InfoData[0] == 'V'
&& InfoData[1] == '2')
                    isV2 = 1;
            }
        SQLFreeStmt (hstmt, SQL_CLOSE);
    }

    vAMPs = 0;
    if (isV2)
        if (SQLExecDirect (hstmt, (SQLCHAR *) "SELECT
HASHAMP(HASHBUCKET(HASHROW(S_SUPPKEY))) FROM SUPPLIER GROUP BY 1
ORDER BY 1", SQL_NTS) != SQL_ERROR)
        {
            if (SQLRowCount (hstmt, &vAMPs) !=
SQL_SUCCESS)
                fprintf (stderr, " Problem in
SQLRowCount!\n");

            if (vAMPs > 0)
                {

```

```

        fprintf (fdetail, " The
system under test has %d virtual AMPs\n", vAMPs);
        fprintf (stdout, " The
system under test has %d virtual AMPs\n", vAMPs);
    }

    SQLFreeStmt (hstmt, SQL_CLOSE);
}

    if (SQLExecDirect (hstmt, (SQLCHAR *) "SELECT COUNT(*)
FROM DBC.SESSIONINFO", SQL_NTS) != SQL_ERROR)
    {
        if (SQLFetch (hstmt) == SQL_SUCCESS)
            if (SQLGetData (hstmt, 1,
SQL_C_SLONG, &crowcount, 4, &actualLen) == SQL_SUCCESS)
            {
                if (crowcount > maxSess)
                {
                    fprintf
(fdetail, " There are %d other sessions logged on to the system
under test\n", crowcount - maxSess);

                    fprintf
(stdout, " There are %d other sessions logged on to the system
under test\n", crowcount - maxSess);
                }
            }
        SQLFreeStmt (hstmt, SQL_CLOSE);
    }

    if (flags & VERBOSE)
        fprintf (stdout, "Submitting SQL Requests to verify
scale factor\n");
    if (SQLExecDirect (hstmt, (SQLCHAR *) "SELECT COUNT(*)
FROM CUSTOMER", SQL_NTS) != SQL_ERROR)
    {
        if (SQLFetch (hstmt) == SQL_SUCCESS)
            if (SQLGetData (hstmt, 1,
SQL_C_SLONG, &crowcount, 4, &actualLen) == SQL_SUCCESS)
            {
                SQLFreeStmt (hstmt,
SQL_CLOSE);
                newSF = ((double)
crowcount) / 150000.0;
                fprintf (stdout, "
Customer table exists with scale factor %g\n", newSF);
            }
    }

```

```

        if ((long) (newSF * 1000
+ 0.5) != (long) (flt_scale * 1000))
            fprintf (stdout,
" Scale factor does not match!\n");
    }
    else
    {
        fprintf (stderr, " Customer table does not
exist or cant be accessed\n");
        SQLError ((SQLHENV) 0, (SQLHDBC) 0, hstmt,
(SQLCHAR *) SqlState, &ErrCode, (SQLCHAR *) ErrMsg, 256,
&ErrMsgLen);
        fprintf (stderr, " %d: %s\n", ErrCode,
ErrMsg);
        SQLFreeStmt (hstmt, SQL_CLOSE);
    }

    if (SQLExecDirect (hstmt, (SQLCHAR *) "SELECT COUNT(*)
FROM PARTTBL", SQL_NTS) != SQL_ERROR)
    {
        if (SQLFetch (hstmt) == SQL_SUCCESS)
            if (SQLGetData (hstmt, 1,
SQL_C_SLONG, &crowcount, 4, &actualLen) == SQL_SUCCESS)
            {
                SQLFreeStmt (hstmt,
SQL_CLOSE);
                newSF = ((double)
crowcount) / 200000.0;
                fprintf (stdout, " Part
table exists with scale factor %g\n", newSF);
                if ((long) (newSF * 1000
+ 0.5) != (long) (flt_scale * 1000))
                    fprintf (stdout,
" Scale factor does not match!\n");
            }
        else
        {
            fprintf (stderr, " Part table does not
exist\n");
            SQLError ((SQLHENV) 0, (SQLHDBC) 0, hstmt,
(SQLCHAR *) SqlState, &ErrCode, (SQLCHAR *) ErrMsg, 256,
&ErrMsgLen);
            fprintf (stderr, " %d: %s\n", ErrCode,
ErrMsg);
            SQLFreeStmt (hstmt, SQL_CLOSE);
        }
    }

```

```

    }

    if (SQLExecDirect (hstmt, (SQLCHAR *) "SELECT COUNT(*)
FROM ORDERTBL", SQL_NTS) != SQL_ERROR)
    {
        if (SQLFetch (hstmt) == SQL_SUCCESS)
            if (SQLGetData (hstmt, 1,
SQL_C_SLONG, &crowcount, 4, &actualLen) == SQL_SUCCESS)
                {
                    SQLFreeStmt (hstmt,
SQL_CLOSE);
                    newSF = ((double)
crowcount) / 1500000.0;
                    fprintf (stdout, " Order
table exists with scale factor %g\n", newSF);
                    if ((long) (newSF * 1000
+ 0.5) != (long) (flt_scale * 1000))
                        fprintf (stdout,
" Scale factor does not match!\n");
                }
            }
        else
            {
                fprintf (stderr, " Order table does not
exist\n");
                SQLError ((SQLHENV) 0, (SQLHDBC) 0, hstmt,
(SQLCHAR *) SqlState, &ErrCode, (SQLCHAR *) ErrMsg, 256,
&ErrMsgLen);
                fprintf (stderr, " %d: %s\n", ErrCode,
ErrMsg);
                SQLFreeStmt (hstmt, SQL_CLOSE);
            }

        return (0);
    }

int
Run_Query_Stream ()
{
    int i;
    ftime (&StartTime);
    fprintf (stdout, " Starting query run for stream %d on
%s\n", snum, ctime (&StartTime.time));

```

```

    fprintf (fdetail, "\n Starting query run for stream %d on
%s\n", snum, ctime (&StartTime.time));

    ftime (&StartTime);

    if (snum >= 0)
        if (optind < ac1)
            for (i = optind; i < ac1; i++)
                {
                    char qname[10];
                    sprintf (qname, "%d", SEQUENCE
(snum, atoi (av1[i])));
                    qsub (qname, flags);
                    Run_Query (SQLRequest,
SEQUENCE (snum, atoi (av1[i])));
                }
            else
                for (i = 1; i <= QUERIES_PER_SET; i++)
                    {
                        char qname[10];
                        sprintf (qname, "%d", SEQUENCE
(snum, i));
                        qsub (qname, flags);
                        Run_Query (SQLRequest,
SEQUENCE (snum, i));
                    }
            else if (optind < ac1)
                for (i = optind; i < ac1; i++)
                    {
                        qsub (av1[i], flags);
                        Run_Query (SQLRequest, atoi
(av1[i]));
                    }
            else
                for (i = 1; i <= QUERIES_PER_SET; i++)
                    {
                        char qname[10];
                        sprintf (qname, "%d", i);
                        qsub (qname, flags);
                        Run_Query (SQLRequest, i);
                    }

    ftime (&StartTime);
    fprintf (stdout, " Finished query run for stream %d on
%s\n", snum, ctime (&StartTime.time));

```

```

        fprintf (fdetail, " Finished query run for stream %d on
%s\n", snum, ctime (&StartTime.time));

        return (0);
}

void
Print_Stream_Summary ()
{
    int i;

    fprintf (fsummary, "\nQuery timing results:\n\n");
    for (i = 1; i <= QUERIES_PER_SET; i++)
        if (QueryTime[i] != 0.0)
            fprintf (fsummary, "Query #%d time was %.2f
seconds.\n", i, QueryTime[i]);
        else
            fprintf (fsummary, "Query #%d was not
run.\n", i);

    if (UF1time != 0.0)
        fprintf (fsummary, "UF1 (insert) time was %.2f
seconds.\n", UF1time);
    else
        fprintf (fsummary, "UF1 (insert) was not run.\n");

    if (UF2time != 0.0)
        fprintf (fsummary, "UF2 (delete) time was %.2f
seconds.\n", UF2time);
    else
        fprintf (fsummary, "UF2 (delete) was not run.\n");
}

void
Run_UF1_Inserts ()
{
    ftime (&StartTime);
    fprintf (stdout, " Starting UF1 insert run for set %d on
%s\n", upd_num, ctime (&StartTime.time));
    fprintf (fdetail, "\n Starting UF1 insert run for set %d
on %s\n", upd_num, ctime (&StartTime.time));

    rowcnt = tdefs[ORDER_LINE].base / 10000 * scale * refresh;

    if (rowcnt != 1500 * scale && flt_scale >= 1.0)
        fprintf (stderr, " Why is rowcnt = %d\n", rowcnt);

```

```

        minrow = rowcnt * upd_num + 1;
        if (flags & VERBOSE)
            fprintf (fdetail, " Inserting from %d for %d txns,
upd_num = %d\n", minrow, rowcnt, upd_num);

        if (use_fastload_for_updates & 1 == 1)
            gen_updates_via_fastload (minrow, rowcnt, upd_num +
1);
        else
            gen_updates (minrow, rowcnt, upd_num + 1);

        upd_num++;
    }

void
Run_UF2_Deletes (int special)
{
    ftime (&StartTime);
    if (special == 0)
        {
            fprintf (stdout, " Starting UF2 delete run
for set %d on %s\n", upd_num - 1, ctime (&StartTime.time));
            fprintf (fdetail, "\n Starting UF2 delete
run for set %d on %s\n", upd_num - 1, ctime (&StartTime.time));
        }

    rowcnt = tdefs[ORDER_LINE].base / 10000 * scale * refresh;
    if (rowcnt != 1500 * scale && flt_scale >= 1.0)
        fprintf (stderr, " Why is rowcnt = %d\n", rowcnt);

    minrow = rowcnt * (upd_num - 1) + 1;
    if (flags & VERBOSE)
        fprintf (fdetail, " Deleting from %d for %d txns,
upd_num = %d\n", minrow, rowcnt, upd_num);

    if (use_fastload_for_updates & 2 == 2)
        gen_deletes_via_fastload (minrow, rowcnt, upd_num,
special);
    else
        gen_deletes (minrow, rowcnt, upd_num, special);
}

int

```

```

main (int ac, char **av)
{
    int i;
    int c;
    int status;
    FILE *ifp;
    char line[LINE_SIZE];
    double QppD;
    double QthD;
    double QphD;
    double sumoflogs;
    double maxtime;
    double mintime;
    struct timeb ThroughputStartTime;
    struct timeb ThroughputEndTime;
    struct timeb difTime;
    char tempstr[200];
    char tempstr2[20];
    char tempstr3[128];

    prog = av[0];
    rowcnt = 0;
    table = (1 << CUST) |
            (1 << SUPP) |
            (1 << NATION) |
            (1 << REGION) |
            (1 << PART_PSUPP) |
            (1 << ORDER_LINE);
    scale = 1;
    flt_scale = (double) 1.0;
    flags = 0;
    updates = 1;
    refresh = UPD_PCT;
    gen_sql = 0;
    gen_rng = 1;
    direct = 1;
    tdefs[ORDER].base *=
        ORDERS_PER_CUST;          /* have to do this after
init */
    tdefs[LINE].base *=
        ORDERS_PER_CUST;          /* have to do this after
init */
    tdefs[ORDER_LINE].base *=
        ORDERS_PER_CUST;          /* have to do this after
init */
    children = 1;
    throughputstreams = 3;

    upd_num = 0;
    QppD = 0.0;
    QthD = 0.0;
    QphD = 0.0;

    ac1 = ac;
    av1 = av;

    process_options (ac, av);
    if (flags & VERBOSE)
        printf ("TPC-H/R Benchmark Driver, (matched to
DBGEN/QGEN Version %d.%d.%d%c)\n",
VERSION, RELEASE ,MODIFICATION,PATCH);
    fflush (stdout);

    if (use_fastload_for_updates == 3)
        maxSess = throughputstreams + 1;

    if ((flags & SEED) == 0)
        if (flags & EVERYTHING)
            num_seeds = throughputstreams + 1;

    if (num_seeds > 1 && num_seeds < throughputstreams + 1 &&
(flags & EVERYTHING))
        {
            fprintf (stderr, " You didn't supply enough
seed values \n");
            exit (1);
        }
    if (num_seeds > 1 && !(flags & EVERYTHING))
        {
            fprintf (stderr, " Supplying multiple seeds
only makes sense if you are also use -e \n");
            exit (1);
        }

    setup ();
    /* have to do this after init */
    tdefs[NATION].base = nations.count;
    tdefs[REGION].base = regions.count;

    for (i = 0; i <= QUERIES_PER_SET; i++)
        SaveSeed[i] = Seed[i].value;

    if (!(flags & DFLT))          /* preturb the RNG */
        {

```

```

    if (!(flags & SEED))
    {
        for (i = MAX_STREAM; i >= 0; i--)
        {
            rndm = rndm =
(long)((unsigned)time(NULL) * DSS_PROC) + i;
            if (rndm < 0)
                rndm +=
2147483647;
            if (rndm == 0)
                rndm = 12345;
            initSeeds[i] = rndm;
        }
    }
    if (num_seeds > 1 && (flags & EVERYTHING))
    {
        for (i = 0; i <
num_seeds; i++)
        {
            printf ("
-- Using %ld as seed for stream %d\n", initSeeds[i], i);
            if
(initSeeds[i] == 0)
            {
                fprintf (stderr, " -- Seeds cannot be zero!\n");
                exit (1);
            }
        }
    }
    else
        printf("-- using %ld as a seed to the RNG\n",
rndm);

    for (i=1; i <= QUERIES_PER_SET; i++)
    {
        Seed[i].value = rndm;
        UnifInt(0L, 100L, i);
        UnifInt(0L, 100L, i);
        UnifInt(0L, 100L, i);
        UnifInt(0L, 100L, i);
        UnifInt(0L, 100L, i);
    }
}

else
    printf("-- using default substitutions\n");

    if (!(flags & DBASE))
    {
        fprintf (stderr, " You must use the -n
option to enter the Teradata logon to use\n");
        exit (1);
    }

    if (!(flags & OUTPUT))
    {
        osuff = malloc (strlen (".") + 1);
        MALLOC_CHECK (osuff);
        strcpy (osuff, ".");
    }

    if (snum >= 0)
        sprintf (tempstr,
"%s/tpcd_stream_%d_detailed_output.txt", osuff, snum);
    else
        sprintf (tempstr, "%s/tpcd_detailed_output.txt",
osuff);
    fdetail = fopen (tempstr, "w");
    if (snum >= 0)
        sprintf (tempstr,
"%s/tpcd_stream_%d_summary_output.txt", osuff, snum);
    else
        sprintf (tempstr, "%s/tpcd_summary_output.txt",
osuff);
    fsummary = fopen (tempstr, "w");

    if (fdetail == NULL || fsummary == NULL)
    {
        fprintf (stderr, " **** Fatal error, cannot
open output files. errno=%d\n", errno);
        if (flags & OUTPUT)
            fprintf (stderr, " **** Probably
directory %s does not exist or is not writable\n", osuff);
        exit (1);
    }
}

```



```

    signal (SIGABRT, stop_proc);
    signal (SIGTERM, stop_proc);
    signal (SIGINT, stop_proc);
#ifdef _WIN32
    signal (SIGBREAK, stop_proc);
#else
    signal (SIGTSTP, stop_proc);
#endif

    fprintf (fdetail, "\n TPC-D Benchmark Execution, detailed
results\n");
    fprintf (fsummary, "\n TPC-D Benchmark Execution, summary
results\n");

    fprintf (fdetail, "\n Using Teradata logon '%s'\n",
db_name);
    fprintf (fsummary, "\n Using Teradata logon '%s'\n",
db_name);

    fprintf (fsummary, " Using %ld as seed \n", rndm);
    fprintf (fdetail, " Using %ld as seed \n", rndm);

    fprintf (fsummary, " Scale factor is %g\n", flt_scale);
    fprintf (fdetail, " Scale factor is %g\n", flt_scale);

    /* Log on all sessions needed */
    Connect_Sessions ();

    /* Test to see the size of the tables */
    if (specialtest == 0 && snum <= 0)
        Verify_Database ();

    for (i = 0; i <= QUERIES_PER_SET; i++)
    {
        QueryTime[i] = 0.0;
    }
    UF1time = 0.0;
    UF2time = 0.0;
    /* Run Power test warm-up, optional */
    /* run update function UF1 once on same query stream! */
    if (updates > 0)
    {

        /* Save Query Seeds */
        for (i = 0; i <= QUERIES_PER_SET; i++)
        {
            tempSeed = SaveSeed[i];
            SaveSeed[i] = Seed[i].value;
            Seed[i].value = tempSeed;
        }

        if (specialtest)
        {
            upd_num = 0;
            system ("prfld");
            system ("prfstat");

            for (i = 1; i < 5; i++)
            {
                setsPerReq = i;
                fprintf (fdetail,
" Testing with setsPerReq = %d \n", setsPerReq);

                upd_num = 1;
                Run_UF2_Deletes

                (1);

                fflush (stdout);
                fflush (stderr);
                fflush (fdetail);
                fflush

                (fsummary);

                sprintf (tempstr,
"prfsnap InsertText&dtxns.log", setsPerReq);

                system (tempstr);
                upd_num = 0;
                Run_UF1_Inserts

                system (tempstr);
                fflush (stdout);
                fflush (stderr);
                fflush (fdetail);
                fflush

                }
            fflush (stdout);
            fflush (stderr);
            fclose (fsummary);
            fclose (fdetail);

            exit (1);
        }
    }
}

```

```

        if ((flags & DFLT) || snum < 0 || flt_scale
< 1.0)
        {
            /* Temp: Pre-delete to clean
up everything from previous failed runs */
            /* normally, this should
delete zero rows */
            upd_num = updates;
            Run_UF2_Deletes (1);
            UF2time = 0.0;
        }
        upd_num = updates - 1;
        fflush (fdetail);
        fflush (fsummary);

/*
 * run set_state to advance the RNG to the start point for insert
generation
 * this will avoid duplicated rows between the inserts and the
original data
 */
        set_state (ORDER, scale, 1, 2, (long *)&i);
        Run_UF1_Inserts ();

        /* Restore Query Seeds, Save Insert seeds */
        for (i = 0; i <= QUERIES_PER_SET; i++)
        {
            tempSeed = SaveSeed[i];
            SaveSeed[i] = Seed[i].value;
            Seed[i].value = tempSeed;
        }

        fflush (fdetail);
        fflush (fsummary);

        if (flags & INIT)          /* init stream with
ifile */
        {
            ifp = fopen (ifile, "r");
            if (ifp == NULL)
            {
                fprintf (stderr, "Failed to
open file '%s'\n", ifile);
                exit (1);
            }
        }
    }
    while (fgets (line, LINE_SIZE, ifp) != NULL)
        fprintf (stdout, "%s", line);
}

/* Run Power test, stream=0! */
Run_Query_Stream ();

/* Restore Insert seeds */
for (i = 0; i <= QUERIES_PER_SET; i++)
{
    tempSeed = SaveSeed[i];
    SaveSeed[i] = Seed[i].value;
    Seed[i].value = tempSeed;
}

fflush (fdetail);
fflush (fsummary);

/* Run update function UF2 once on the same query stream!
*/
if (updates > 0)
    if (snum < 0 || (flags & DFLT))
        Run_UF2_Deletes (1);
    else
        Run_UF2_Deletes (0);

    fprintf (stdout, " Stream %d finished on %s\n", snum,
ctime (&StartTime.time));
    fprintf (fdetail, "\n Stream %d finished on %s\n", snum,
ctime (&StartTime.time));

Print_Stream_Summary ();

fflush (fdetail);
fflush (fsummary);

if (snum > 0)
    if (flags & EVERYTHING)
    {
        Disconnect_Sessions ();

        fclose (fsummary);
        fclose (fdetail);
    }
}

```

```

        return (0);
    }

    maxtime = 0.0;
    mintime = 9e50;
    for (i = 1; i <= QUERIES_PER_SET; i++)
    {
        if (maxtime < QueryTime[i])
            maxtime = QueryTime[i];
        if (mintime > QueryTime[i])
            mintime = QueryTime[i];
    }
    if (maxtime > 0.0)
        if ((maxtime / mintime) > 1000.0)
        {
            fprintf (fsummary, " Ratio of max
query time to min query time over 1000, special processing in
effect\n");
            fprintf (fdetail, " Ratio of max
query time to min query time over 1000, special processing in
effect\n");

            for (i = 1; i < QUERIES_PER_SET; i++)
                if (QueryTime[i] < maxtime /
1000.0)
                    QueryTime[i] = maxtime /
1000.0;
        }

    if (maxtime > 0.0)
    {
        sumoflogs = 0.0;
        for (i = 1; i <= QUERIES_PER_SET; i++)
            sumoflogs += log (QueryTime[i]);
        sumoflogs += log (UF1time) + log (UF2time);
        QppD = (3600.0 / exp (sumoflogs / 24.0)) *
flt_scale;

        fprintf (fsummary, "\n QppD@%.1fGB =
%.2f\n", flt_scale, QppD);
        fprintf (fdetail, "\n QppD@%.1fGB = %.2f\n",
flt_scale, QppD);
    }

    /* Power test complete! */

```

```

    if (snum == 0)
        if (flags & EVERYTHING)
        {
            /*note: nothing is legal between
power test and throughput test */
            /* Run throughput in parallel with a
single update stream */
            /*note: stream 1 to s, where s=
#streams */

            ftime (&StartTime);
            ThroughputStartTime.time =
StartTime.time;
            ThroughputStartTime.millitm =
StartTime.millitm;

            fprintf (stdout, " Beginning
throughput test using %d query streams on %s\n",
throughputstreams, ctime (&StartTime.time));
            fprintf (fdetail, "\n Beginning
throughput test using %d query streams on %s\n",
throughputstreams, ctime (&StartTime.time));

            for (c = 0; c < throughputstreams;
c++)
                {
                    sprintf (tempstr, "%d",
c + 1);
                    sprintf (tempstr2, "%f",
flt_scale);
                    if (c < num_seeds)
                        sprintf
(tempstr3, "%ld", initSeeds[c + 1]);
                    else
                        sprintf
(tempstr3, "%ld", rndm);

                    #if (defined(WIN32)&&!defined(_POSIX_))
                        pids[c] = _spawnl
(_P_NOWAIT, av[0], av[0], "-p", tempstr, "-s", tempstr2, "-U",
"0", "-r", tempstr3, "-n", db_name,
"-o", osuff,
"-v",
(char *) NULL);
                    #endif
                }
        }

```

```

        fprintf (stdout, "
Spawned task, pid is %d\n", pids[c]);
        if (pids[c] == -1)
        {
            fprintf
(stderr, "Child query stream not created %d\n", errno);
            exit (-1);
        }
    #else
        pids[c] = SPAWN ();
        if (pids[c] == -1)
        {
            fprintf
(stderr, "Child query stream not created");
            exit (-1);
        }
        else if (pids[c] == 0)
            /* CHILD */
            {
                execl
(av[0], av[0], "-p", tempstr, "-s", tempstr2, "-U", "0", "-r",
tempstr3, "-n", db_name,
"-o", osuff,
(char *) NULL);
                fprintf
(stderr, " Could not execl to start the new task %d\n", errno);
                exit (1);
            }
        else
            if (flags & VERBOSE) /*
                fprintf (stdout,
                    "waiting...");
                if (flags & VERBOSE)
                    fprintf (stdout,
                        fflush (stderr);
                        fflush (fdetail);
                        fflush (fsummary);
                    c = throughputstreams;
                    while (c)
                    {
                        #if (defined(WIN32)&&!defined(_POSIX_))
                            i = _cwait (&status,
                                pids[c - 1], _WAIT_CHILD);
                            if (i == -1 &&
                                throughputstreams)
                                {
                                    if (errno
                                        == ECHILD)
                                        fprintf (stderr, "Could not wait on pid %d\n", pids[c -
                                            1]);
                                    else if
                                        (errno == EINTR)
                                        fprintf (stderr, "Process %d stopped abnormally\n", pids[c
                                            - 1]);
                                    else if
                                        (errno == EINVAL)
                                        fprintf (stderr, "Program bug\n");
                                }
                            else
                                {
                                    fprintf
                                        (stdout, "Process %d: STOPPED\n", pids[c - 1]);
                                }
                            #else
                                i = wait (&status);
                                if (i == -1 &&
                                    throughputstreams)
                                    {
                                        fprintf
                                            (stderr, "We lost one\n");
                                        exit (-2);
                                    }
                                if (status & 0xFF)
                                    {
                                        if (status
                                            & 0xFF == 0117)
                                            printf ("Process %d: STOPPED\n", i);
                                        else
                                            printf ("Process %d: rcvd signal %d\n",

```

```

        i, status & 0x7F);
    }
#endif
        c--;
    }
    ftime (&StartTime);
    fprintf (stdout, " Throughput query
streams ended on %s, starting updates\n", ctime
(&StartTime.time));
    fprintf (fdetail, "\n Throughput
query streams ended on %s, starting updates\n", ctime
(&StartTime.time));

    if (updates > 0)
    {
        for (c = 0; c <
throughputstreams; c++)
        {
            Run_UF1_Inserts
();
            if (snum < 0 ||
(flags & DFLT))
                Run_UF2_Deletes (1);
            else
                Run_UF2_Deletes (0);
        }
    }
    ftime (&ThroughputEndTime);
    difTime.time = ThroughputEndTime.time
- ThroughputStartTime.time;
    if (ThroughputEndTime.millitm >=
ThroughputStartTime.millitm)
    {
        difTime.millitm =
(unsigned short) (ThroughputEndTime.millitm -
ThroughputStartTime.millitm);
    }
    else
    {

```

```

        difTime.millitm =
(unsigned short) ((1000 + ThroughputEndTime.millitm) -
ThroughputStartTime.millitm);
        difTime.time -= 1;
    };
    Throughputtime = difTime.time +
difTime.millitm / 1000.0 + 0.005;
    fprintf (stdout, " Throughput test
ended on %s\n", ctime (&ThroughputEndTime.time));
    fprintf (fdetail, "\n Throughput
test ended on %s\n", ctime (&ThroughputEndTime.time));

    fprintf (fdetail, "\n Throughput
test took %.2f seconds\n", Throughputtime);

    if (updates > 0)
    {
        QthD =
((throughputstreams * 22.0 * 3600.0) / Throughputtime) *
flt_scale;
        fprintf (fsummary, "\n
QthD@%.1fGB = %.2f\n", flt_scale, QthD);
        fprintf (fdetail, "\n
QthD@%.1fGB = %.2f\n", flt_scale, QthD);

        QphD = sqrt (QppD *
QthD);
        fprintf (fsummary, "\n
QphD@%.1fGB = %.2f\n", flt_scale, QphD);
        fprintf (fdetail, "\n
QphD@%.1fGB = %.2f\n", flt_scale, QphD);
    }
}

    if (flags & TERMINATE)
        /* terminate stream with
tfile */

```

```
    {
        ifp = fopen (tfile, "r");
        if (ifp == NULL)
            {
                fprintf (stderr, "Failed to
open terminate file '%s'\n",
                                tfile);
                exit (1);
            }
        while (fgets (line, LINE_SIZE, ifp) != NULL)
            fprintf (stdout, "%s", line);
    }

    fflush (fdetail);
    fflush (fsummary);

    Disconnect_Sessions ();

    fclose (fsummary);
    fclose (fdetail);

    return (0);
}
```

Implementation specific layer

To simplify the source to the tpccdriver, the details of calling Teradata CLIv2 was hidden in this layer of code designed to look similar to ISO CLI or ODBC CLI.

Tdatsql.h

```
/*
 * tdatsql.h This file just includes the standard ODBC or ISO CLI
 * header files (see ISO/IEC 9075-3:1995, or the Microsoft ODBC
 * SDK
 *
 */

#undef FAR
#define FAR

typedef void * HWND;
#include <sqltypes.h>
#include <sql.h>

/* Two Teradata-specific functions, because this made my life
   easier than using the standard ISO CLI or ODBC function */

SQLRETURN SQL_API SQLSetUsingParcel(
    SQLHSTMT hstmt,
    char * uptr,
    long ulen);
```

```
SQLRETURN SQL_API SQLWaitForAny(
    SQLHENV henv,
    SQLHSTMT * lphstmt,
    int * stmtno);
```

tsqlt.h

```
/*
 * tsqlt.h
 * Teradata SQL Type codes are different from ODBC SQL Type
 * codes,
 * so we need to translate them. This translates from Teradata
 * form
 * to ODBC form. The Teradata form also contains the Nullable
 * information,
 * so we extract that at the same time.
 */

void TranslateSQLType( SWORD * SqlType, SWORD * Nullable, SWORD
OldSqlType );

/* This translates the other way, from ODBC SQL Type to Teradata
SQL Type. */
#ifdef __cplusplus
extern "C"
#endif
SWORD UnTranslateSQLType( SWORD SqlType );
```

tsqlt.c

```
/*
 * TSQLT.C
 * Teradata SQL Type codes are different from ODBC SQL Type
 * codes,
 * so we need to translate them. This translates from Teradata
 * form
 * to ODBC form. The Teradata form also contains the Nullable
 * information,
 * so we extract that at the same time.
 */
#include "tdatsql.h"
```

```

void TranslateSQLType( SWORD * SqlType, SWORD * Nullable, SWORD
OldSqlType )
{
    switch ( OldSqlType )
    {
case 384:                // IBM Date
case 385:
case 570:                // DBC date (old style)?
case 571:
case 572:
case 573:
case 752:                // DBC date
case 753:
        *SqlType = SQL_DATE;
        break;

case 388:
case 389:
        *SqlType = SQL_TIME;
        break;

case 392:
case 393:
        *SqlType = SQL_TIMESTAMP;
        break;

case 448:
case 449:
        *SqlType = SQL_VARCHAR;
        break;

case 452:
case 453:
        *SqlType = SQL_CHAR;
        break;

case 456:
case 457:
        *SqlType = SQL_LONGVARCHAR;
        break;

case 460:
case 461:
        *SqlType = SQL_VARCHAR;    // with zero byte at end.
        break;

case 468:                // GRAPHIC
case 469:
        *SqlType = SQL_CHAR; // ?
        break;

case 464:                // VARGRAPHIC
case 465:
case 472:                // LONG VARGRAPHIC
case 473:
        *SqlType = SQL_VARCHAR;    // ?
        break;

case 480:
case 481:
        *SqlType = SQL_FLOAT;
        break;

case 484:
case 485:
        *SqlType = SQL_DECIMAL;
        break;

case 496:
case 497:
        *SqlType = SQL_INTEGER;
        break;

case 500:
case 501:
        *SqlType = SQL_SMALLINT;
        break;

case 576:
case 577:
case 756:
case 757:
        *SqlType = SQL_TINYINT;
        break;

case 588:
case 589:
case 692:
case 693:
        *SqlType = SQL_BINARY;
        break;

case 582:

```



```

    case 583:
    case 688:
    case 689:
    case 594:
    case 595:
    case 696:
    case 697:
        *SqlType = SQL_VARBINARY;
        break;

    default:
        ;
    }
    if ( ( OldSqlType & 1 ) == 1 )
        *Nullable = SQL_NULLABLE;
    else
        *Nullable = SQL_NO_NULLS;
    }

// This translates the other way, from ODBC SQL Type to Teradata
SQL Type.
// SDK 2.0 introduced signed/unsigned LONG, SHORT & TINYINT.
Signed types map
// to the same SqlType as SDK 1.0, but for unsigned they have to
map to a larger
// type. This was added to fix GCA list #1011, where parameter
passing
// (binding) of signed/unsigned types was not working.
SWORD UnTranslateSQLType( SWORD SqlType )
{
    switch ( SqlType )
    {
    case SQL_DATE:
    case SQL_TIMESTAMP:
        return 753;        // DBC Date
        break;

    case SQL_VARCHAR:
        return 449;
        break;

    case SQL_CHAR:
        return 453;
        break;

    case SQL_LONGVARCHAR:
        return 457;
    }
}

break;

case SQL_REAL:
    return 481;    // Teradata does not have single prec
real.

case SQL_FLOAT:
case SQL_DOUBLE:
case SQL_TIME:
    return 481;

case SQL_DECIMAL:
// Rudy Ezquerro. GCA list #1011, [un]signed types not
bound properly
// SQL_C_ULONG = SQL_INTEGER + SQL_UNSIGNED_OFFSET
// Best match for ULONG is DECIMAL(10,0) ==> 8 bytes
case SQL_C_ULONG:
    return 485;

case SQL_INTEGER:
// SQL_C_SLONG = SQL_INTEGER + SQL_SIGNED_OFFSET
// SQL_C_USHORT = SQL_SMALLINT + SQL_UNSIGNED_OFFSET
case SQL_C_SLONG:
case SQL_C_USHORT:
    return 497;

case SQL_SMALLINT:
// SQL_C_SSHORT = SQL_SMALLINT + SQL_SIGNED_OFFSET
// SQL_C_UTINYINT = SQL_TINYINT + SQL_UNSIGNED_OFFSET
case SQL_C_SSHORT:
case SQL_C_UTINYINT:
    return 501;

case SQL_TINYINT:
case SQL_BIT:
// SQL_C_STINYINT = SQL_TINYINT + SQL_SIGNED_OFFSET
case SQL_C_STINYINT:
    return 757;

case SQL_BINARY:
    return 693;

case SQL_VARBINARY:
    return 689;

default:
    return SqlType;
}

```

```
}  
}
```

tdatsql.c

```
/*  
 * tdatsql.c This is an interface layer that gives a very  
 primitive  
 * ODBC style (or ISO CLI style) call level interface look on top  
 of  
 * Teradata's standard CLIV2. This allows the tpcddriver.c to be  
 * a bit more readable and cleaner, and we hide the CLIV2 stuff  
 in  
 * hear. Only enough of ODBC/ISO CLI is implemented to run just  
 * what we need for TPC-D, although this code could be used for  
 other  
 * things.  
 * The idea is that someday, when we have either an ANSI X/Open  
 * or ISO CLI on Unix, or an ODBC on Unix, we can just rip out  
 this  
 * code and throw it away, and use the ISO or ODBC cli.  
 */
```

```
#include <stdio.h>  
#include <stdlib.h>  
#include <string.h>  
#include <time.h>  
#include <limits.h>  
#include <math.h>  
#include <sys/types.h>  
#include <sys/timeb.h>  
#if (defined(_POSIX_) || !defined(WIN32))  
#include <unistd.h>  
#define INLINE _Inline  
#ifndef snprintf  
#define snprintf(w,x,y,z) sprintf(w,y,z)  
#endif  
#else  
#include <process.h>  
/*#include <signal.h> */  
#include <errno.h>  
/*  
    #pragma warning(disable:4201)  
    #pragma warning(disable:4214)  
    #pragma warning(disable:4514)
```

```
#define WIN32_LEAN_AND_MEAN  
#define NOATOM  
#define NOGDICAPMASKS  
#define NOMETAFILE  
#define NOMINMAX  
#define NOMSG  
#define NOOPENFILE  
#define NORASTEROPS  
#define NOSCROLL  
#define NOSOUND  
#define NOSYSTEMETRICS  
#define NOTEXTMETRIC  
#define NOWH  
#define NOCOMM  
#define NOKANJI  
#define NOMCX  
#include <windows.h>  
#pragma warning(default:4201)  
#pragma warning(default:4214)  
*/  
#define INLINE _inline  
#ifndef itoa  
#define itoa _itoa  
#endif  
#ifndef sprintf  
#define sprintf _sprintf  
#endif  
#include <ctype.h>  
#include <time.h>  
#include <assert.h>  
#include "tdatsql.h"  
#include "tsqlt.h"  
  
#include <coptypes.h>  
#include <dbcarea.h>  
#include <coperr.h>  
#include <dbcerr.h>  
#include <parcel.h>  
  
#ifndef __min  
#define __min(a,b) ((a) < (b)) ? (a) : (b)  
#endif  
  
#define MALLOC_CHECK(var) \  
    if ((var) == NULL) \  
    { \  
        \  
    } \  
}
```

```

    fprintf(stderr, "Malloc failed at %s:%d\n", \
        __FILE__, __LINE__); \
    exit(1);\
}
#define DBCINTN      496
#define DBCINTI      497
#define DBCSINTN     500
#define DBCSINTI     501
#define DBCDECN      484
#define DBCDECI      485
#define DBCFLOATN    480
#define DBCFLOATI    481
#define DBCVCHARN    448
#define DBCVCHARI    449
#define DBCCHARN     452
#define DBCCHARI     453
#define DBCLCHARN    456
#define DBCLCHARI    457
#define DBCDATEN     752
#define DBCDATEI     753
#define DBCBINTN     756
#define DBCBINTI     757
#define DBCVBYTEN    688
#define DBCVBYTEI    689
#define DBCBYTEN     692
#define DBCBYTEI     693
#define DBCLBYTEN    696
#define DBCLBYTEI    697

#define LINE_SIZE 512

char *CLICTXT = NULL;

long hstmt_next_num = 1;

/*
    INLINE
    void DR_VStr(char * x)
    {
        short len;
        len=(short)strlen(x);
        memcpy(pl,&len,2);
        pl += 2;
        memcpy(pl,x,len);
        pl += len;

```

```

    }
    INLINE
    void DR_Str(char * x,size_t y)
    {
        memset(pl,' ',y);
        memcpy(pl,x,strlen(x));
        pl+=y;
    }
    INLINE
    void DR_Money(long x)
    {
        long temp = 0;
        memcpy(pl,&x,4);
        pl += 4;
        memcpy(pl,&temp,4);
        pl += 4;
    }
    INLINE
    void DR_Char(char x)
    {
        *pl = x; pl++;
    }
    INLINE
    void DR_Date(char * x)
    {
        long tempdate;
        int y,m,d;
        y=atoi(x);
        m=atoi(x+5);
        d=atoi(x+8);
        if (y<1800 || y>2200 || m<1 || m>12 || d<1 || d>31)
        {
            fprintf(stderr," Help! I can't interpret the date %s\n",x);
            exit(1);
        }
        tempdate=(y-1900)*10000+m*100+d;
        memcpy(pl,&tempdate,4);
        pl += 4;
    }
*/
typedef struct DBC
    {
        Int32 SessionId;

```

```

        Int32 ReqId;
        short numResultCols;
        long RowCount;
        char *p;
        unsigned short Async;
        unsigned short busy;
        long NativeError;
        char ErrorMessage[512];
        struct CliDataInfoType DataInfo;
        struct DBCAREA dbcarea;
    }
DBC;

typedef struct STMT
{
    DBC *lpdbc;
    long hstmt_num;
}
STMT;

SQLRETURN SQL_API
SQLAllocConnect (
                SQLHENV henv,
                SQLHDBC FAR * phdbc)
{
    Int32 ReturnCode;
    struct DBC *lpdbc;
    lpdbc = malloc (sizeof (struct DBC));
    lpdbc->SessionId = 0;
    lpdbc->ReqId = 0;
    lpdbc->numResultCols = 0;
    lpdbc->RowCount = 0;
    lpdbc->p = NULL;
    lpdbc->Async = 0;
    lpdbc->busy = 0;
    lpdbc->NativeError = (-999);
    lpdbc->ErrorMessage[0] = '\0';
    memset (&lpdbc->DataInfo, 0, sizeof (lpdbc->DataInfo));
    memset (&lpdbc->dbcarea, 0, sizeof (lpdbc->dbcarea));

    strncpy (lpdbc->dbcarea.eyecatcher, "DBCAREA", 8);

        lpdbc->dbcarea.total_len = sizeof (struct DBCAREA);
        DBCHINI (&ReturnCode, CLICTXT, &lpdbc->dbcarea);
        if (ReturnCode != 0)
            {
                fprintf (stderr, " The DBCHINI return code
was %ld\n", ReturnCode);
                strncpy (lpdbc->ErrorMessage, lpdbc-
>dbcarea.msg_text, lpdbc->dbcarea.msg_len);
                lpdbc->ErrorMessage[lpdbc->dbcarea.msg_len]
= '\0';
                lpdbc->NativeError = ReturnCode;
                return (SQL_ERROR);
            }

        assert (lpdbc->dbcarea.total_len == sizeof (struct
DBCAREA));
        strncpy (lpdbc->dbcarea.eyecatcher, "DBCAREA", 8);

        lpdbc->dbcarea.req_buf_len = 32700;
        lpdbc->dbcarea.resp_buf_len = 32700;
        lpdbc->dbcarea.max_num_sess = 100;
        lpdbc->dbcarea.resp_mode = 'I';
        lpdbc->dbcarea.keep_resp = 'N';
        lpdbc->dbcarea.use_presence_bits = 'N';
        lpdbc->dbcarea.wait_for_resp = 'Y';
        lpdbc->dbcarea.loc_mode = 'Y';
        lpdbc->dbcarea.var_len_req = 'N';
        lpdbc->dbcarea.var_len_fetch = 'N';
        lpdbc->dbcarea.save_resp_buf = 'N';
        lpdbc->dbcarea.two_resp_bufs = 'N';
        lpdbc->dbcarea.req_proc_opt = 'E';
        lpdbc->dbcarea.change_opts = 'Y';
        lpdbc->dbcarea.wait_for_resp = 'Y';
        lpdbc->dbcarea.ret_time = 'Y';
        lpdbc->dbcarea.parcel_mode = 'Y';
        lpdbc->dbcarea.tell_about_crash = 'Y'; /* Set crash
options */
        lpdbc->dbcarea.wait_across_crash = 'N'; /* Set
crash options */
        lpdbc->dbcarea.msg_security = 'N';
        *phdbc = (SQLHDBC) lpdbc;
        return (SQL_SUCCESS);
    }

SQLRETURN SQL_API
SQLAllocEnv (
                SQLHENV FAR * phenv)

```

```

{
    return (SQL_SUCCESS);
}

SQLRETURN SQL_API
SQLAllocStmt (
    SQLHDBC hdbc,
    SQLHSTMT FAR * phstmt)
{
    struct STMT *lpstmt;
    lpstmt = malloc (sizeof (struct STMT));
    lpstmt->lpdbc = (DBC *) hdbc;
    lpstmt->hstmt_num = hstmt_next_num++;
    *phstmt = (SQLHSTMT) lpstmt;
    return (SQL_SUCCESS);
}

SQLRETURN SQL_API
SQLSetStmtOption (
    SQLHSTMT hstmt,
    SQLUSMALLINT fOption,
    SQLUIINTEGER vParam)
{
    if (fOption == SQL_ASYNC_ENABLE)
        ((STMT *) hstmt)->lpdbc->Async = (USHORT) vParam;

    if (((STMT *) hstmt)->lpdbc->Async != SQL_ASYNC_ENABLE_ON)
        ((STMT *) hstmt)->lpdbc->dbcarea.wait_for_resp =
'Y';
    return (SQL_SUCCESS);
}

SQLRETURN SQL_API
SQLColAttributes (
    SQLHSTMT hstmt,
    SQLUSMALLINT icol,
    SQLUSMALLINT fDescType,
    SQLPOINTER rgbDesc,
    SQLSMALLINT cbDescMax,
    SQLSMALLINT FAR * pcbDesc,
    SQLINTEGER FAR * pfDesc)
{
    SWORD SType;
    SWORD Nullable;
    char temp[20];
    sprintf (temp, "%hd", icol);

    if (icol > 0)
    {
        TranslateSQLType (&SType, &Nullable, ((STMT
*) hstmt)->lpdbc->DataInfo.InfoVar[icol - 1].SQLType);
    }
    switch (fDescType)
    {
        case SQL_COLUMN_LABEL:
        case SQL_COLUMN_NAME:
        {
            strcpy (rgbDesc, "Col");
            strcat (rgbDesc, temp);
            *pcbDesc = (SQLSMALLINT) strlen
((char *) rgbDesc);
            return (SQL_SUCCESS);
        }
        case SQL_COLUMN_TYPE:
            *pfDesc = (long) SType;
            break;
        case SQL_COLUMN_LENGTH:
            *pfDesc = ((STMT *) hstmt)->lpdbc-
>DataInfo.InfoVar[icol - 1].SQLLen;
            switch (SType)
            {
                case SQL_BIT:
                case SQL_TINYINT:
                    *pfDesc = 1;
                    break;
                case SQL_SMALLINT:
                    *pfDesc = 2;
                    break;
                case SQL_INTEGER:
                    *pfDesc = 4;
                    break;
                case SQL_BIGINT:
                    *pfDesc = 20;
                    break;
                case SQL_REAL:
                    *pfDesc = 4;
                    break;
                case SQL_FLOAT:
                case SQL_DOUBLE:
                    *pfDesc = 8;
                    break;
                case SQL_DATE:
                case SQL_TIME:
            }
    }
}

```

```

                *pfDesc = 6;
                break;
            case SQL_TIMESTAMP:
                *pfDesc = 16;
                break;
            case SQL_NUMERIC:
            case SQL_DECIMAL:
                *pfDesc = ((STMT *) hstmt)-
>lpdbc->DataInfo.InfoVar[icol - 1].SQLLen + 2;
                break;
            default:
                *pfDesc = ((STMT *) hstmt)-
>lpdbc->DataInfo.InfoVar[icol - 1].SQLLen;
        }

        break;
    case SQL_COLUMN_PRECISION:
        switch (SType)
        {
            case SQL_BIT:
                *pfDesc = 1;
                break;
            case SQL_TINYINT:
                *pfDesc = 3;
                break;
            case SQL_SMALLINT:
                *pfDesc = 5;
                break;
            case SQL_INTEGER:
                *pfDesc = 10;
                break;
            case SQL_BIGINT:
                *pfDesc = 19;
                break;
            case SQL_REAL:
                *pfDesc = 7;
                break;
            case SQL_FLOAT:
            case SQL_DOUBLE:
                *pfDesc = 15;
                break;
            case SQL_DATE:
                *pfDesc = 10;
                break;
            case SQL_TIME:
                *pfDesc = 8;
                break;

```

```

            case SQL_TIMESTAMP:
                *pfDesc = 23;
                break;
            case SQL_DECIMAL:
                *pfDesc = ((STMT *) hstmt)-
>lpdbc->DataInfo.InfoVar[icol - 1].SQLLen / 256;
                break;
            default:
                *pfDesc = ((STMT *) hstmt)-
>lpdbc->DataInfo.InfoVar[icol - 1].SQLLen;
        }
        break;
    case SQL_COLUMN_SCALE:
        *pfDesc = 0;
        if (SType == SQL_DECIMAL)
            *pfDesc = ((STMT *) hstmt)->lpdbc-
>DataInfo.InfoVar[icol - 1].SQLLen & 0x00FF;
        break;
    case SQL_COLUMN_DISPLAY_SIZE:
        switch (SType)
        {
            case SQL_BIT:
                *pfDesc = 1;
                break;
            case SQL_TINYINT:
                *pfDesc = 4;
                break;
            case SQL_SMALLINT:
                *pfDesc = 6;
                break;
            case SQL_INTEGER:
                *pfDesc = 11;
                break;
            case SQL_BIGINT:
                *pfDesc = 20;
                break;
            case SQL_REAL:
                *pfDesc = 13;
                break;
            case SQL_FLOAT:
            case SQL_DOUBLE:
                *pfDesc = 22;
                break;
            case SQL_DATE:
                *pfDesc = 10;
                break;
            case SQL_TIME:

```

```

                *pfDesc = 8;
                break;
            case SQL_TIMESTAMP:
                *pfDesc = 23;
                break;
            case SQL_BINARY:
            case SQL_VARBINARY:
            case SQL_LONGVARBINARY:
                *pfDesc = ((STMT *) hstmt)-
>lpdbc->DataInfo.InfoVar[icol - 1].SQLLen * 2;
            case SQL_NUMERIC:
            case SQL_DECIMAL:
                *pfDesc = ((STMT *) hstmt)-
>lpdbc->DataInfo.InfoVar[icol - 1].SQLLen / 256 + 2;
                break;

                default:
                    *pfDesc = *pfDesc = ((STMT *)
hstmt)->lpdbc->DataInfo.InfoVar[icol - 1].SQLLen;
                }

                break;

            case SQL_COLUMN_NULLABLE:
                *pfDesc = Nullable;
                break;

            default:
                return (SQL_ERROR);
        }

        return (SQL_SUCCESS);
    }

SQLRETURN SQL_API
SQLRowCount (
                SQLHSTMT hstmt,
                SQLINTEGER FAR * pcrow)
{
    *pcrow = ((STMT *) hstmt)->lpdbc->RowCount;
    return (SQL_SUCCESS);
}

SQLRETURN SQL_API
SQLNumResultCols (
                SQLHSTMT hstmt,
                SQLSMALLINT FAR * pccol)

```

```

    {
        *pccol = (short) ((STMT *) hstmt)->lpdbc->numResultCols;
        return (SQL_SUCCESS);
    }
SQLRETURN SQL_API
SQLError (
                SQLHENV henv,
                SQLHDBC hdbc,
                SQLHSTMT hstmt,
                SQLCHAR FAR * szSqlState,
                SQLINTEGER FAR * pfNativeError,
                SQLCHAR FAR * szErrorMsg,
                SQLSMALLINT cbErrorMsgMax,
                SQLSMALLINT FAR * pcbErrorMsg)
{
    DBC *lpdbc;
    if (hstmt != 0)
        lpdbc = ((STMT *) hstmt)->lpdbc;
    else
        lpdbc = (DBC *) hdbc;
    *pfNativeError = lpdbc->NativeError;
    if (lpdbc->NativeError == 0)
        strcpy ((char *) szSqlState, "00000");
    else
        strcpy ((char *) szSqlState, "S1000");
    strcpy ((char *) szErrorMsg, lpdbc->ErrorMessage);
    *pcbErrorMsg = (SQLSMALLINT) strlen (lpdbc->ErrorMessage);
    if (lpdbc->NativeError == 0)
        return (SQL_NO_DATA_FOUND);
    else
        return (SQL_SUCCESS);
}

SQLRETURN SQL_API
SQLFreeStmt (
                SQLHSTMT hstmt,
                SQLUSMALLINT fOption)
{
    Int32 ReturnCode;
    ((STMT *) hstmt)->lpdbc->dbcarea.i_sess_id = ((STMT *)
hstmt)->lpdbc->SessionId;
    ((STMT *) hstmt)->lpdbc->dbcarea.i_req_id = ((STMT *)
hstmt)->lpdbc->ReqId;
    ((STMT *) hstmt)->lpdbc->dbcarea.func = DBFERQ;
    DBCHCL (&ReturnCode, CLICTXT, &((STMT *) hstmt)->lpdbc-
>dbcarea);
}

```

```

((STMT *) hstmt)->lpdbc->dbcarea.using_data_len = 0;
((STMT *) hstmt)->lpdbc->dbcarea.using_data_ptr = NULL;

if (ReturnCode != 0)
    {
        if (ReturnCode != 305) /* Session not
found */
            {
                fprintf (stderr, " The ERQ
return code was %ld\n", ReturnCode);
                strncpy (((STMT *) hstmt)-
>lpdbc->ErrorMessage, ((STMT *) hstmt)->lpdbc->dbcarea.msg_text,
((STMT *) hstmt)->lpdbc->dbcarea.msg_len);
                ((STMT *) hstmt)->lpdbc-
>ErrorMessage[ ((STMT *) hstmt)->lpdbc->dbcarea.msg_len] = '\0';
                ((STMT *) hstmt)->lpdbc-
>NativeError = ReturnCode;
                return (SQL_ERROR);
            }
        }
    return (SQL_SUCCESS);
}

```

```

SQLRETURN SQL_API
SQLGetData (

```

```

    SQLHSTMT hstmt,
    SQLUSMALLINT icol,
    SQLSMALLINT fctype,
    SQLPOINTER rgbValue,
    SQLINTEGER cbValueMax,
    SQLINTEGER FAR * pcbValue)
{
    SWORD SType;
    SWORD Nullable;
    char *p2;
    static char tempStr[512];
    double tempFloat;
    unsigned long tempUInt;
    long tempInteger;
    short tempSmallint;

```

```

((STMT *) hstmt)->lpdbc->NativeError = 0;

    if ((int) icol > (int) ((STMT *) hstmt)->lpdbc-
>numResultCols)
        return (SQL_NO_DATA_FOUND);
    assert (((STMT *) hstmt)->lpdbc->numResultCols > 0);
    assert (icol > 0 && (int) icol <= (int) ((STMT *) hstmt)-
>lpdbc->numResultCols);
    assert (((STMT *) hstmt)->lpdbc->dbcarea.fet_ret_data_len
> 0);
    if (icol == 1)
        {
            ((STMT *) hstmt)->lpdbc->p = ((struct
CliRecordType *) ((STMT *) hstmt)->lpdbc->dbcarea.fet_data_ptr)-
>Body;
            assert (((STMT *) hstmt)->lpdbc->p != NULL);

            ((STMT *) hstmt)->lpdbc->p += (((STMT *)
hstmt)->lpdbc->numResultCols + 7) / 8); /* Skip Indicator bytes
*/
        }
    p2 = ((struct CliRecordType *) ((STMT *) hstmt)->lpdbc-
>dbcarea.fet_data_ptr)->Body +
        ((STMT *) hstmt)->lpdbc->dbcarea.fet_ret_data_len;

    if (icol > 0)
        {
            TranslateSQLType (&SType, &Nullable, ((STMT
*) hstmt)->lpdbc->DataInfo.InfoVar[icol - 1].SQLType);
        }
    assert (((STMT *) hstmt)->lpdbc->p != NULL);
    assert (p2 != NULL);
    assert (0 == strcmp (((STMT *) hstmt)->lpdbc-
>dbcarea.eyecatcher, "DBCAREA"));
    assert (((STMT *) hstmt)->lpdbc->dbcarea.total_len ==
sizeof (struct DBCAREA));

    if (((STMT *) hstmt)->lpdbc->p >= p2)
        return (SQL_NO_DATA_FOUND);

    if (fctype == SQL_C_DEFAULT &&
        (SType == SQL_CHAR || SType == SQL_VARCHAR))
        fctype = SQL_C_CHAR;

    if (fctype == SQL_C_DEFAULT &&

```



```

        (SType == SQL_INTEGER))
        fCType = SQL_C_SLONG;

        assert (fCType == SQL_C_CHAR || fCType == SQL_C_LONG ||
fCType == SQL_C_SLONG);

        if (pcbValue != NULL)
            *pcbValue = 0;

        switch (((STMT *) hstmt)->lpdbc->DataInfo.InfoVar[icol -
1].SQLType & 0xFFFE)
        {
            case 496:                /* integer */
                memcpy (&tempInteger, ((STMT *) hstmt)-
>lpdbc->p, 4);
                ((STMT *) hstmt)->lpdbc->p += 4;
                break;
            case 500:                /* smallint */
                memcpy (&tempSmallint, ((STMT *) hstmt)-
>lpdbc->p, 4);
                tempInteger = tempSmallint;
                ((STMT *) hstmt)->lpdbc->p += 2;
                break;
            case 484:                /* decimal */

                if (((STMT *) hstmt)->lpdbc-
>DataInfo.InfoVar[icol - 1].SQLLen / 256 > 9)
                    {
                        memcpy (&tempUInt, ((STMT *)
hstmt)->lpdbc->p, 4);
                        tempFloat = tempUInt;

                        memcpy (&tempInteger, ((STMT
*) hstmt)->lpdbc->p + 4, 4);
                        tempFloat += tempInteger *
4294967296.0;

                        ((STMT *) hstmt)->lpdbc->p +=
8;
                    }
                else
                    {
                        if (((STMT *) hstmt)->lpdbc-
>DataInfo.InfoVar[icol - 1].SQLLen / 256 > 4)
                            {

```

```

                                memcpy
                                (&tempInteger, ((STMT *) hstmt)->lpdbc->p, 4);
                                tempFloat =
                                ((STMT *) hstmt)-
                                >lpdbc->p += 4;
                                }
                                else
                                {
                                    if (((STMT *)
hstmt)->lpdbc->DataInfo.InfoVar[icol - 1].SQLLen / 256 > 2)
                                        {
                                            memcpy (&tempSmallint, ((STMT *) hstmt)->lpdbc->p, 2);
                                            tempInteger = tempSmallint;
                                            tempFloat = tempSmallint;
                                            ((STMT *) hstmt)->lpdbc->p += 2;
                                        }
                                    else
                                        {
                                            tempFloat = *((STMT *) hstmt)->lpdbc->p;
                                            ((STMT *) hstmt)->lpdbc->p += 1;
                                        }
                                }
                                }

                                switch (((STMT *) hstmt)->lpdbc-
>DataInfo.InfoVar[icol - 1].SQLLen & 0x00FF)
                                {
                                    case 15:
                                        tempFloat /=
1000000000000000.0;
                                        break;
                                    case 14:
                                        tempFloat /=
1000000000000000.0;
                                        break;
                                    case 13:
                                        tempFloat /= 10000000000000.0;
                                        break;
                                    case 12:
                                        tempFloat /= 1000000000000.0;

```

```

                break;
case 11:
    tempFloat /= 100000000000.0;
    break;
case 10:
    tempFloat /= 10000000000.0;
    break;
case 9:
    tempFloat /= 1000000000.0;
    break;
case 8:
    tempFloat /= 100000000.0;
    break;
case 7:
    tempFloat /= 10000000.0;
    break;
case 6:
    tempFloat /= 1000000.0;
    break;
case 5:
    tempFloat /= 100000.0;
    break;
case 4:
    tempFloat /= 10000.0;
    break;
case 3:
    tempFloat /= 1000.0;
    break;
case 2:
    tempFloat /= 100.0;
    break;
case 1:
    tempFloat /= 10.0;
    break;
default:;
}

                break;
case 480:
    memcpy (&tempFloat, ((STMT *) hstmt)->lpdbc-
>p, 8);
    ((STMT *) hstmt)->lpdbc->p += 8;
    break;
case 452:
    tempSmallint = ((STMT *) hstmt)->lpdbc-
>DataInfo.InfoVar[icol - 1].SQLLen;
                memcpy (tempStr, (char *) ((STMT *) hstmt)-
>lpdbc->p, __min (tempSmallint, 511));
                tempStr[__min (tempSmallint, 511)] = '\0';
                ((STMT *) hstmt)->lpdbc->p += tempSmallint;
                break;
                case 448:
                memcpy (&tempSmallint, ((STMT *) hstmt)-
>lpdbc->p, 2);
                memcpy (tempStr, (char *) (((STMT *) hstmt)-
>lpdbc->p + 2), __min (tempSmallint, 511));
                tempStr[__min (tempSmallint, 511)] = '\0';
                ((STMT *) hstmt)->lpdbc->p += tempSmallint +
2;
                break;
                case 752:
                memcpy (&tempInteger, ((STMT *) hstmt)-
>lpdbc->p, 4);
                tempInteger += 19000000;
                ((STMT *) hstmt)->lpdbc->p += 4;
                break;
                case 756:
                tempSmallint = (short) *(char *) ((STMT *)
hstmt)->lpdbc->p;
                tempInteger = tempSmallint;
                ((STMT *) hstmt)->lpdbc->p += 1;
                break;
                default:
                *pcbValue = snprintf (rgbValue, cbValueMax,
"%S", "????");
                ((STMT *) hstmt)->lpdbc->p += ((STMT *)
hstmt)->lpdbc->DataInfo.InfoVar[icol - 1].SQLLen;
                break;
            }
            if (fCType == SQL_C_CHAR)
            {
                ((char *) rgbValue)[0] = '\0';
                switch (((STMT *) hstmt)->lpdbc-
>DataInfo.InfoVar[icol - 1].SQLType & 0xFFFE)
                {
                    case 496:
                        /* integer */
                        *pcbValue = snprintf
(rgbValue, cbValueMax, "%10ld", tempInteger);
                        break;
                    case 500:
                        /* smallint */
                        *pcbValue = snprintf
(rgbValue, cbValueMax, "%6hd", tempSmallint);

```

```

                break;
            case 484:                /* decimal */
                *pcbValue = snprintf
#ifdef WIN32
                (rgbValue, cbValueMax, "%15.*f", ((STMT *) hstmt)->lpdbc-
                >DataInfo.InfoVar[icol - 1].SQLLen & 0x00FF, tempFloat);
            #else
                *pcbValue = sprintf (rgbValue,
                "%15.*f", ((STMT *) hstmt)->lpdbc->DataInfo.InfoVar[icol -
                1].SQLLen & 0x00FF, tempFloat);
            #endif
                break;
            case 480:                /* float */
                *pcbValue = snprintf
                (rgbValue, cbValueMax, "%15g", tempFloat);
                break;
            case 452:
            case 448:
                *pcbValue = snprintf
                (rgbValue, cbValueMax, "%s", tempStr);
                break;
            case 752:
#ifdef WIN32
                *pcbValue = snprintf
                (rgbValue, cbValueMax, "%04ld-%02d-%02d", tempInteger / 10000,
                (tempInteger / 100) % 100, tempInteger % 100);
            #else
                *pcbValue = sprintf (rgbValue,
                "%04ld-%02d-%02d", tempInteger / 10000, (tempInteger / 100) %
                100, tempInteger % 100);
            #endif
                break;
            case 756:
                *pcbValue = snprintf
                (rgbValue, cbValueMax, "%4hd", tempSmallint);
                break;
            default:
                *pcbValue = snprintf
                (rgbValue, cbValueMax, "%s", "????");
                break;
        }
        if (*pcbValue < 0)
        {
            fprintf (stderr, " sprintf
failed!\n");
        }
        *pcbValue = strlen (rgbValue);
    }
}

else if (fCType == SQL_C_LONG || fCType == SQL_C_SLONG)
{
    if (pcbValue != NULL)
        *pcbValue = 4;
    switch (((STMT *) hstmt)->lpdbc-
    >DataInfo.InfoVar[icol - 1].SQLType & 0xFFFE)
    {
        case 496:
        case 500:
        case 752:
        case 756:                /* integer,
smallint, byteint, date */
            memcpy (rgbValue,
            &tempInteger, 4);
            break;
        case 484:
        case 480:                /* decimal, float
*/
            tempInteger = (long)
tempFloat;
            memcpy (rgbValue,
            &tempInteger, 4);
            break;
        default:
            tempInteger = 0;
            memcpy (rgbValue,
            &tempInteger, 4);
            break;
    }
}

return (SQL_SUCCESS);
}

SQLRETURN SQL_API
SQLFetch (
        SQLHSTMT hstmt)
{
    Int32 ReturnCode;
    long ActivityCount;
}

```

```

struct CliSuccessType *successPtr;
struct CliFailureType *failurePtr;
struct CliDataInfoType *datainfoPtr;

((STMT *) hstmt)->lpdbc->p = NULL;

((STMT *) hstmt)->lpdbc->dbcarea.i_sess_id = ((STMT *)
hstmt)->lpdbc->SessionId;
((STMT *) hstmt)->lpdbc->dbcarea.i_req_id = ((STMT *)
hstmt)->lpdbc->ReqId;
((STMT *) hstmt)->lpdbc->dbcarea.change_opts = 'y';

assert (((STMT *) hstmt)->lpdbc->SessionId > 0);
assert (((STMT *) hstmt)->lpdbc->ReqId > 0);

assert (0 == strcmp (((STMT *) hstmt)->lpdbc-
>dbcarea.eyecatcher, "DBCAREA"));
assert (((STMT *) hstmt)->lpdbc->dbcarea.total_len ==
sizeof (struct DBCAREA));

((STMT *) hstmt)->lpdbc->dbcarea.func = DBFFET;
DBCHCL (&ReturnCode, CLICTXT, &((STMT *) hstmt)->lpdbc-
>dbcarea);
if (((STMT *) hstmt)->lpdbc->Async && ReturnCode == 305)
{
    fprintf (stderr, " Fetch still executing?
The FETCH return code was %ld\n", ReturnCode);
    return (SQL_STILL_EXECUTING);
}
if (ReturnCode != 0)
{
    fprintf (stderr, " The FETCH return code was
%ld\n", ReturnCode);
    strncpy (((STMT *) hstmt)->lpdbc-
>ErrorMessage, ((STMT *) hstmt)->lpdbc->dbcarea.msg_text, ((STMT
*) hstmt)->lpdbc->dbcarea.msg_len);
    ((STMT *) hstmt)->lpdbc->ErrorMessage[ ((STMT
*) hstmt)->lpdbc->dbcarea.msg_len] = '\0';
    ((STMT *) hstmt)->lpdbc->NativeError =
ReturnCode;
    return (SQL_ERROR);
}

assert (((STMT *) hstmt)->lpdbc->dbcarea.i_sess_id ==
((STMT *) hstmt)->lpdbc->SessionId);
assert (((STMT *) hstmt)->lpdbc->dbcarea.i_req_id ==
((STMT *) hstmt)->lpdbc->ReqId);

switch (((STMT *) hstmt)->lpdbc-
>dbcarea.fet_parcel_flavor)
{
    case PclSUCCESS:
        successPtr = (struct CliSuccessType *)
((STMT *) hstmt)->lpdbc->dbcarea.fet_data_ptr;
        memcpy (&ActivityCount, successPtr-
>ActivityCount, 4);
        ((STMT *) hstmt)->lpdbc->RowCount =
ActivityCount;
        ((STMT *) hstmt)->lpdbc->numResultCols =
successPtr->FieldCount;
        ((STMT *) hstmt)->lpdbc->busy = 0;
        ((STMT *) hstmt)->lpdbc->dbcarea.change_opts
= 'Y';
        ((STMT *) hstmt)->lpdbc-
>dbcarea.wait_for_resp = 'Y';
        break;
    case PclERROR:
    case PclFAILURE:
        ((STMT *) hstmt)->lpdbc->busy = 0;
        ((STMT *) hstmt)->lpdbc->dbcarea.change_opts
= 'Y';
        ((STMT *) hstmt)->lpdbc-
>dbcarea.wait_for_resp = 'Y';
        failurePtr = (struct CliFailureType *)
((STMT *) hstmt)->lpdbc->dbcarea.fet_data_ptr;
        ((STMT *) hstmt)->lpdbc->NativeError =
failurePtr->Code;
        strncpy (((STMT *) hstmt)->lpdbc-
>ErrorMessage, failurePtr->Msg, (size_t) (failurePtr->Length));
        ((STMT *) hstmt)->lpdbc-
>ErrorMessage[ (failurePtr->Length)] = '\0';

        fprintf (stderr, " Failure: %d %s\n", ((STMT
*) hstmt)->lpdbc->NativeError, ((STMT *) hstmt)->lpdbc-
>ErrorMessage);

        SQLFreeStmt (hstmt, SQL_CLOSE);
        assert (0 == strcmp (((STMT *) hstmt)-
>lpdbc->dbcarea.eyecatcher, "DBCAREA"));

```

```

        assert (((STMT *) hstmt)->lpdbc-
>dbcarea.total_len == sizeof (struct DBCAREA));
        return (SQL_ERROR);
        /*break; */
    case PclDATAINFO:
        datainfoPtr = (struct CliDataInfoType *)
((STMT *) hstmt)->lpdbc->dbcarea.fet_data_ptr;
        assert (datainfoPtr->FieldCount == ((STMT *)
hstmt)->lpdbc->numResultCols);
        memcpy (&((STMT *) hstmt)->lpdbc->DataInfo,
datainfoPtr, sizeof (struct CliDataInfoType));
        assert (((STMT *) hstmt)->lpdbc-
>DataInfo.FieldCount == ((STMT *) hstmt)->lpdbc->numResultCols);
        assert (0 == strcmp (((STMT *) hstmt)-
>lpdbc->dbcarea.eyecatcher, "DBCAREA"));
        assert (((STMT *) hstmt)->lpdbc-
>dbcarea.total_len == sizeof (struct DBCAREA));
        break;
    case PclENDSTATEMENT:
        ((STMT *) hstmt)->lpdbc->NativeError = 0;
        ((STMT *) hstmt)->lpdbc->busy = 0;
        return (SQL_NO_DATA_FOUND);
        /*break; */
    case PclENDREQUEST:
        SQLFreeStmt (hstmt, SQL_CLOSE);
        ((STMT *) hstmt)->lpdbc->NativeError = 0;
        return (SQL_NO_DATA_FOUND);
        /*break; */
    case PclRECORD:
        ((STMT *) hstmt)->lpdbc->NativeError = 0;
        ((STMT *) hstmt)->lpdbc->p = ((struct
CliRecordType *) ((STMT *) hstmt)->lpdbc->dbcarea.fet_data_ptr)-
>Body;
        assert (((STMT *) hstmt)->lpdbc->p != NULL);

        ((STMT *) hstmt)->lpdbc->p += (((STMT *)
hstmt)->lpdbc->numResultCols + 7) / 8); /* Skip Indicator bytes
*/

        return (SQL_SUCCESS);
        /*break; */

    default:
        fprintf (stderr, " Dont know this parcel
type\n");

        ((STMT *) hstmt)->lpdbc->NativeError = (-1);
        return (SQL_ERROR);
}

return (SQL_SUCCESS);
}

SQLRETURN SQL_API
SQLMoreResults (
                SQLHSTMT hstmt)
{
    int rc;
    ((STMT *) hstmt)->lpdbc->dbcarea.i_sess_id = ((STMT *)
hstmt)->lpdbc->SessionId;
    ((STMT *) hstmt)->lpdbc->dbcarea.i_req_id = ((STMT *)
hstmt)->lpdbc->ReqId;
    ((STMT *) hstmt)->lpdbc->dbcarea.wait_for_resp = 'Y';
    ((STMT *) hstmt)->lpdbc->dbcarea.change_opts = 'Y';
    ((STMT *) hstmt)->lpdbc->numResultCols = 0;
    ((STMT *) hstmt)->lpdbc->RowCount = 0;

    if ((rc = SQLFetch (hstmt)) != SQL_SUCCESS)
        return ((SQLRETURN) rc); /* The success parcel */
    assert (0 == strcmp (((STMT *) hstmt)->lpdbc-
>dbcarea.eyecatcher, "DBCAREA"));
    assert (((STMT *) hstmt)->lpdbc->dbcarea.total_len ==
sizeof (struct DBCAREA));
    ((STMT *) hstmt)->lpdbc->dbcarea.using_data_len = 0;
    ((STMT *) hstmt)->lpdbc->dbcarea.using_data_ptr = NULL;
    while (((STMT *) hstmt)->lpdbc->dbcarea.fet_parcel_flavor
== PclRECORD)
        if ((rc = SQLFetch (hstmt)) != SQL_SUCCESS)
            return ((SQLRETURN) rc);
        if (((STMT *) hstmt)->lpdbc->dbcarea.fet_parcel_flavor ==
PclENDSTATEMENT)
            if ((rc = SQLFetch (hstmt)) != SQL_SUCCESS)
                return ((SQLRETURN) rc);
            if (((STMT *) hstmt)->lpdbc->dbcarea.fet_parcel_flavor ==
PclENDREQUEST)
                return (SQL_NO_DATA_FOUND);

        if ((rc = SQLFetch (hstmt)) != SQL_SUCCESS)
            {
                if (rc != SQL_NO_DATA_FOUND)
                    return ((SQLRETURN) rc); /* Data
Info parcel or End */
            }
        assert (0 == strcmp (((STMT *) hstmt)->lpdbc-
>dbcarea.eyecatcher, "DBCAREA"));
}

```

```

    assert (((STMT *) hstmt)->lpdbc->dbcarea.total_len ==
sizeof (struct DBCAREA));
    return (SQL_SUCCESS);
}

SQLRETURN SQL_API
SQLConnect (
    SQLHDBC hdbc,
    SQLCHAR FAR * szDSN,
    SQLSMALLINT cbDSN,
    SQLCHAR FAR * szUID,
    SQLSMALLINT cbUID,
    SQLCHAR FAR * szAuthStr,
    SQLSMALLINT cbAuthStr)
{
    SQLHSTMT fakehstmt;

    Int32 ReturnCode;
    static char UserInfo[250];

    assert (((DBC *) hdbc)->dbcarea.total_len == sizeof
(struct DBCAREA));
    strncpy (((DBC *) hdbc)->dbcarea.eyecatcher, "DBCAREA",
8);

    ((DBC *) hdbc)->dbcarea.req_buf_len = 8192;
    ((DBC *) hdbc)->dbcarea.resp_buf_len = 32700;
    ((DBC *) hdbc)->dbcarea.max_num_sess = 16;
    ((DBC *) hdbc)->dbcarea.resp_mode = 'I';
    ((DBC *) hdbc)->dbcarea.keep_resp = 'N';
    ((DBC *) hdbc)->dbcarea.use_presence_bits = 'N';
    ((DBC *) hdbc)->dbcarea.wait_for_resp = 'Y';
    ((DBC *) hdbc)->dbcarea.loc_mode = 'Y';
    ((DBC *) hdbc)->dbcarea.var_len_req = 'N';
    ((DBC *) hdbc)->dbcarea.var_len_fetch = 'N';
    ((DBC *) hdbc)->dbcarea.save_resp_buf = 'N';
    ((DBC *) hdbc)->dbcarea.two_resp_bufs = 'N';
    ((DBC *) hdbc)->dbcarea.req_proc_opt = 'E';
    ((DBC *) hdbc)->dbcarea.change_opts = 'Y';
    ((DBC *) hdbc)->dbcarea.wait_for_resp = 'Y';
    ((DBC *) hdbc)->dbcarea.ret_time = 'Y';
    ((DBC *) hdbc)->dbcarea.parcel_mode = 'Y';
    ((DBC *) hdbc)->dbcarea.tell_about_crash = 'Y'; /*
Set crash options */
    ((DBC *) hdbc)->dbcarea.wait_across_crash = 'N'; /*
Set crash options */

```

```

    ((DBC *) hdbc)->dbcarea.msg_security = 'N';

    strcpy (UserInfo, (char *) szDSN);
    strcat (UserInfo, "/");
    strcat (UserInfo, (char *) szUID);
    strcat (UserInfo, ",");
    strcat (UserInfo, (char *) szAuthStr);
    ((DBC *) hdbc)->dbcarea.logon_ptr = (char *) UserInfo;
    ((DBC *) hdbc)->dbcarea.logon_len = strlen (UserInfo);
    ((DBC *) hdbc)->dbcarea.run_ptr = NULL;
/*
    dbcarea.run_ptr = (char *) connect_struct;
    dbcarea.run_len = sizeof (struct CliCONNECTType);
    dbcarea.inter_ptr = CharSet;
    dbcarea.charset_type = CharSetType; */
    ((DBC *) hdbc)->dbcarea.sess_2pc_mode = 'N';
    ((DBC *) hdbc)->dbcarea.using_data_len = 0;
    ((DBC *) hdbc)->dbcarea.using_data_ptr = NULL;

    ((DBC *) hdbc)->dbcarea.func = DBFCON;
    DBCHCL (&ReturnCode, CLICTXT, &((DBC *) hdbc)->dbcarea);
    if (ReturnCode == EM_NETCONN)
    {
        fprintf (stderr, "Teradata system is down,
waiting to logon...\n");
        strcpy (((DBC *) hdbc)->ErrorMessage,
"Teradata system is down.");
        return (-1);
    }
    //while (ReturnCode == EM_NETCONN) {
    //    /* loop until connected or some other error then
DBC down. */
    //    dbcarea.func = DBFCON;
    //    DBCHCL (&ReturnCode, CLICTXT, dbcarea);
    // }
    if (ReturnCode != 0)
    {
        fprintf (stderr, " The logon return code was
%d\n", ReturnCode);
        strcpy (((DBC *) hdbc)->ErrorMessage, ((DBC
*) hdbc)->dbcarea.msg_text, ((DBC *) hdbc)->dbcarea.msg_len);
        ((DBC *) hdbc)->ErrorMessage[((DBC *) hdbc)-
>dbcarea.msg_len] = '\0';
        ((DBC *) hdbc)->NativeError = ReturnCode;
    }

```

```

        ((DBC *) hdbc)->dbcarea.i_sess_id = ((DBC *) hdbc)-
>dbcarea.o_sess_id;
        ((DBC *) hdbc)->dbcarea.i_req_id = ((DBC *) hdbc)-
>dbcarea.o_req_id;
        ((DBC *) hdbc)->SessionId = ((DBC *) hdbc)-
>dbcarea.o_sess_id;
        ((DBC *) hdbc)->ReqId = ((DBC *) hdbc)->dbcarea.o_req_id;

fakehstmt = (SQLHSTMT) malloc (sizeof (struct STMT));
((STMT *) fakehstmt)->lpdbc = (DBC *) hdbc;
if (SQLFetch (fakehstmt) == SQL_ERROR)
    return (SQL_ERROR);

assert (0 == strcmp (((DBC *) hdbc)->dbcarea.eyecatcher,
"DBCAREA"));
assert (((DBC *) hdbc)->dbcarea.total_len == sizeof
(struct DBCAREA));

//fprintf(stderr,"Session %d, Req %d logged on\n",((DBC
*)hdbc)->dbcarea.o_sess_id,((DBC *)hdbc)->dbcarea.o_req_id);
return (SQL_SUCCESS);
}

```

```

SQLRETURN SQL_API
SQLExecDirect (

```

```

                SQLHSTMT hstmt,
                SQLCHAR FAR * szSqlStr,
                SQLINTEGER cbSqlStr)
{
    Int32 ReturnCode;
    int rc;

    assert (((STMT *) hstmt)->lpdbc->SessionId > 0);
    assert (((STMT *) hstmt)->lpdbc->ReqId >= 0);
    assert (0 == strcmp (((STMT *) hstmt)->lpdbc-
>dbcarea.eyecatcher, "DBCAREA"));
    assert (((STMT *) hstmt)->lpdbc->dbcarea.total_len ==
sizeof (struct DBCAREA));

    ((STMT *) hstmt)->lpdbc->dbcarea.i_sess_id = ((STMT *)
hstmt)->lpdbc->SessionId;
    ((STMT *) hstmt)->lpdbc->dbcarea.i_req_id = ((STMT *)
hstmt)->lpdbc->ReqId;

```

```

        ((STMT *) hstmt)->lpdbc->dbcarea.change_opts = 'Y';
        if (((STMT *) hstmt)->lpdbc->busy == 0)
            {
                assert (szSqlStr != NULL);

                ((STMT *) hstmt)->lpdbc-
>dbcarea.wait_for_resp = 'Y';
                ((STMT *) hstmt)->lpdbc->dbcarea.req_ptr =
(char *) szSqlStr;
                ((STMT *) hstmt)->lpdbc->dbcarea.req_len =
strlen ((char *) szSqlStr);

                ((STMT *) hstmt)->lpdbc->numResultCols = 0;
                ((STMT *) hstmt)->lpdbc->RowCount = 0;
                ((STMT *) hstmt)->lpdbc->dbcarea.token =
(Int32) hstmt;

                ((STMT *) hstmt)->lpdbc->dbcarea.func =
DBFIRQ;
                DBCHCL (&ReturnCode, CLICTXT, &((STMT *)
hstmt)->lpdbc->dbcarea);
                if (ReturnCode != 0)
                    {
                        fprintf (stderr, " The IRQ
return code was %ld\n", ReturnCode);
                        strncpy (((STMT *) hstmt)-
>lpdbc->ErrorMessage, ((STMT *) hstmt)->lpdbc->dbcarea.msg_text,
((STMT *) hstmt)->lpdbc->dbcarea.msg_len);
                        ((STMT *) hstmt)->lpdbc-
>ErrorMessage[((STMT *) hstmt)->lpdbc->dbcarea.msg_len] = '\0';
                        ((STMT *) hstmt)->lpdbc-
>NativeError = ReturnCode;
                        return (SQL_ERROR);
                    }
                ((STMT *) hstmt)->lpdbc->busy = 1;
                //fprintf(stderr,"SQL request running on
Session %d, Req %d\n",((STMT*)hstmt)->lpdbc-
>dbcarea.o_sess_id,((STMT*)hstmt)->lpdbc->dbcarea.o_req_id);
                if (((STMT *) hstmt)->lpdbc->Async ==
SQL_ASYNC_ENABLE_ON)
                    return SQL_STILL_EXECUTING;
            }
}

```

```

        //assert(((STMT *)hstmt)->lpdbc->SessionId ==
dbcarea.o_sess_id);
        ((STMT *) hstmt)->lpdbc->SessionId = ((STMT *) hstmt)-
>lpdbc->dbcarea.o_sess_id;
        ((STMT *) hstmt)->lpdbc->ReqId = ((STMT *) hstmt)->lpdbc-
>dbcarea.o_req_id;
        assert (((STMT *) hstmt)->lpdbc->SessionId > 0);
        assert (((STMT *) hstmt)->lpdbc->ReqId >= 0);
        assert (0 == strcmp (((STMT *) hstmt)->lpdbc-
>dbcarea.eyecatcher, "DBCAREA"));
        assert (((STMT *) hstmt)->lpdbc->dbcarea.total_len ==
sizeof (struct DBCAREA));

        ((STMT *) hstmt)->lpdbc->dbcarea.i_sess_id = ((STMT *)
hstmt)->lpdbc->SessionId;
        ((STMT *) hstmt)->lpdbc->dbcarea.i_req_id = ((STMT *)
hstmt)->lpdbc->ReqId;

        rc = SQLFetch (hstmt);
        if (rc != SQL_SUCCESS)
            return ((SQLRETURN) rc); /* The success parcel */
        assert (0 == strcmp (((STMT *) hstmt)->lpdbc-
>dbcarea.eyecatcher, "DBCAREA"));
        assert (((STMT *) hstmt)->lpdbc->dbcarea.total_len ==
sizeof (struct DBCAREA));
        ((STMT *) hstmt)->lpdbc->dbcarea.using_data_len = 0;
        ((STMT *) hstmt)->lpdbc->dbcarea.using_data_ptr = NULL;
        ((STMT *) hstmt)->lpdbc->dbcarea.wait_for_resp = 'Y';
        ((STMT *) hstmt)->lpdbc->dbcarea.change_opts = 'Y';
        ((STMT *) hstmt)->lpdbc->busy = 0;

        rc = SQLFetch (hstmt);
        if (rc != SQL_SUCCESS)
            {
                if (rc == SQL_NO_DATA_FOUND)
                    return SQL_SUCCESS;
                return ((SQLRETURN) rc); /* Data Info
parcel or End */
            }
        assert (0 == strcmp (((STMT *) hstmt)->lpdbc-
>dbcarea.eyecatcher, "DBCAREA"));
        assert (((STMT *) hstmt)->lpdbc->dbcarea.total_len ==
sizeof (struct DBCAREA));
        return (SQL_SUCCESS);
}

```

```

SQLRETURN SQL_API
SQLDisconnect (
                                SQLHDBC hdbc)
{
    Int32 ReturnCode;

    assert (((DBC *) hdbc)->SessionId > 0);
    assert (((DBC *) hdbc)->ReqId >= 0);
    assert (0 == strcmp (((DBC *) hdbc)->dbcarea.eyecatcher,
"DBCAREA"));
    assert (((DBC *) hdbc)->dbcarea.total_len == sizeof
(struct DBCAREA));
    ((DBC *) hdbc)->dbcarea.i_sess_id = ((DBC *) hdbc)-
>SessionId;
    ((DBC *) hdbc)->dbcarea.i_req_id = ((DBC *) hdbc)->ReqId;
    ((DBC *) hdbc)->dbcarea.using_data_len = 0;
    ((DBC *) hdbc)->dbcarea.using_data_ptr = NULL;
    ((DBC *) hdbc)->dbcarea.func = DBFDSC;
    DBCHCL (&ReturnCode, CLICTXT, &((DBC *) hdbc)->dbcarea);
    if (ReturnCode != 0)
        {
            fprintf (stderr, " The LOGOFF return code
was %ld\n", ReturnCode);
            strncpy (((DBC *) hdbc)->ErrorMessage, ((DBC
*) hdbc)->dbcarea.msg_text, ((DBC *) hdbc)->dbcarea.msg_len);
            ((DBC *) hdbc)->ErrorMessage[((DBC *) hdbc)-
>dbcarea.msg_len] = '\0';
            ((DBC *) hdbc)->NativeError = ReturnCode;
            return (SQL_ERROR);
        }

    /* DBCHCLN (&ReturnCode, CLICTXT);
    if (ReturnCode!=0) {
        fprintf(stderr," The HCLN return code was %ld\n",ReturnCode);
        strncpy(((DBC *)hdbc)-
>ErrorMessage,dbcarea.msg_text,dbcarea.msg_len);
        ((DBC *)hdbc)->ErrorMessage[dbcarea.msg_len]='\0';
        ((DBC *)hdbc)->NativeError= ReturnCode;
        return(SQL_ERROR);
    }
    */

    return (SQL_SUCCESS);
}

SQLRETURN SQL_API

```



```

SQLSetUsingParcel (
    SQLHSTMT hstmt,
    char *uptr,
    long ulen)
{
    ((STMT *) hstmt)->lpdbc->dbcarea.using_data_ptr = uptr;
    ((STMT *) hstmt)->lpdbc->dbcarea.using_data_len = ulen;
    return (SQL_SUCCESS);
}

SQLRETURN SQL_API
SQLWaitForAny (
    SQLHENV henv,
    SQLHSTMT * lphstmt,
    int *stmtno)
{
    Int32 ReturnCode;
    Int32 sessionid;
    Int32 reqtoken;
    STMT *lpstmt;
    int rc;
    sessionid = 0;
    reqtoken = 0;
    *lphstmt = SQL_NULL_HSTMT;
    *stmtno = -1;

    DBCHWAT (&ReturnCode, CLICTXT, &sessionid, &reqtoken);
    if (ReturnCode == NOACTIVE)
    {
        fprintf (stderr, " DBCHWAT says nothing was
active %ld, %ld\n", sessionid, reqtoken);
        return SQL_NO_DATA_FOUND;
    }
    if (ReturnCode != 0)
    {
        fprintf (stderr, " DBCHWAT returned %d\n",
ReturnCode);
        return SQL_ERROR;
    }

    lpstmt = (STMT *) reqtoken;
    if (lpstmt->lpdbc->SessionId != sessionid)
        fprintf (stderr, " DBCHWAT returned the wrong
session?\n");
    *lphstmt = (SQLHSTMT) lpstmt;
    *stmtno = lpstmt->hstmt_num;

    if (lpstmt->lpdbc->busy)
    {
        //fprintf(stderr, " DBCHWAT returned a busy
session completion on %d\n",sessionid);
        lpstmt->lpdbc->dbcarea.wait_for_resp = 'Y';
        lpstmt->lpdbc->dbcarea.change_opts = 'Y';
        rc = SQLExecDirect ((SQLHSTMT) lpstmt,
(SQLCHAR *) NULL, 0);
        if (rc != SQL_SUCCESS && rc !=
SQL_SUCCESS_WITH_INFO)
        {
            //fprintf (stderr, "
Completion of request failed! rc = %d\n", rc);
            if (rc == SQL_NO_DATA_FOUND)
                rc = SQL_SUCCESS;
            return (SQLRETURN) rc;
        }
    }

    //fprintf(stderr, " ***DBCHWAT returned an idle session
%d\n",sessionid);
    return SQL_SUCCESS;
}

```