



ORACLE[®]

TPC Benchmark[™] D Full Disclosure Report

Sun Starfire[™] Enterprise[™] 10000 Using
Oracle8i[™]

Submitted for Review

February 12, 1999

TPC Benchmark D Full Disclosure Report

Release 8i

First Printing

© 1999 Sun Microsystems, Inc.
2550 Garcia Avenue, Mountain View, California 94043-1100 U.S.A.

All rights reserved. This product and related documentation are protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or related documentation may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any.

RESTRICTED RIGHTS LEGEND: Use, duplication, or disclosure by the United States Government is subject to the restrictions set forth in DFARS 252.227-7013 (c)(1)(ii) and FAR 52.227-19, Rights in Technical Data and Computer Software (October 1988).

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

TRADEMARKS

Sun, Sun Microsystems, the Sun logo, Enterprise 10000, Starfire, SMCC, the SMCC logo, SunSoft, the SunSoft logo, Solaris, SunOS, OpenWindows, DeskSet, ONC, and NFS are trademarks or registered trademarks of Sun Microsystems, Inc. All other product names mentioned herein are the trademarks of their respective owners.

All SPARC trademarks, including the SCD Compliant Logo, are trademarks or registered trademarks of SPARC International, Inc. SPARCstation, SPARCserver, SPARCengine, SPARCworks, and SPARCcompiler are licensed exclusively to Sun Microsystems, Inc. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK® and Sun™ Graphical User Interfaces were developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

TPC-D Benchmark™ is a trademark of the Transaction Processing Performance Council.

Oracle8i, SQL*DBA, SQL*oader, SQL*Net and SQL*Plus are registered trademarks of Oracle Corporation.

Veritas is a registered trademark of Veritas Corporation.

THIS PUBLICATION IS PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS PUBLICATION COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THE PUBLICATION. SUN MICROSYSTEMS, INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS PUBLICATION AT ANY TIME.

Sun Microsystems, Inc., believes that the information in this document is accurate as of its publication date. The information in this document is subject to change without notice. Sun Microsystems, Inc., assumes no responsibility for any errors that may appear in this document.

The pricing information in this document is believed to accurately reflect prices in effect on February 12, 1999. However, Sun Microsystems, Inc. and Oracle Corporation provide no warranty on the pricing information in this document.

The performance information in this document is for guidance only. System performance is highly dependent on many factors including system hardware, system and user software, and user application characteristics. Customer applications must be carefully evaluated before estimating performance. Sun Microsystems, Inc., does not warrant or represent that a user can or will achieve a similar performance. No warranty on system performance or price/performance is expressed or implied in this document.

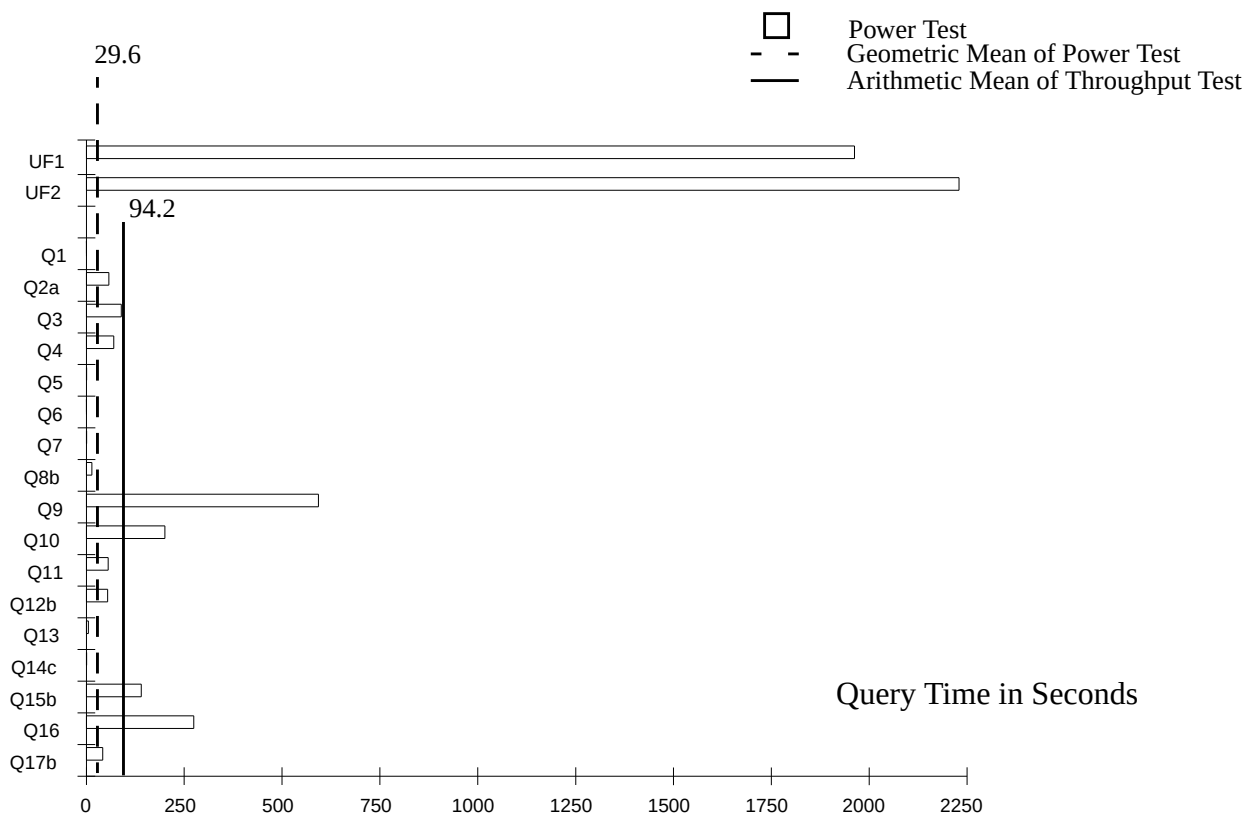


Sun Starfire Enterprise 10000 using Oracle8i

TPC-D 1.3.1

Report Date:
February 12, 1999

Total System Cost	TPC-D Power	TPC-D Throughput	Price/Performance	
\$10,161,744	121,824.7 QppD@1000GB	10,566.3 QthD@1000GB	\$283 per QphD@1000GB	
Database Size	Database Manager	Operating System	Other Software	Availability Date
1,000GB	Oracle8i v 8.1.5.2	Solaris 2.6	Veritas Volume Manager 2.6	August 1, 1999 See Note on next page



Database Load Time = 46 hours 59 minutes

Disk Size/Database Size = 9.81

RAID : N

System Configuration:

- 64 400 Mhz UltraSPARC 64-bit CPUs each with 4MB Cache
- 60 GBytes memory
- 7 Sun StorEdge 1200GB A5200 with 6 x 22 x 9.1 GB drives each
- 4 SPARCStorage A3000 with 35 x 9.1 GB drives each
- 2 SPARCStorage RSM with 7 x 9.1GB drives each



Sun Starfire Enterprise 10000 using Oracle8i

TPC-D 1.3.1

Report Date:
February 12, 1999

Description	Part Number	Source	Unit Price	Qty	Ext. Price	5 Yr. Maint.
Server Hardware						
E10000 Base Cabinet	E10000-3	2	165,000	1	165,000	475,669
System Board, unpopulated	2760A	2	48,750	16	780,000	818,196
CPU, 400MHz, 4MB L2\$	2570A	2	12,750	64	816,000	
Memory Board, unpopulated	7025A	2	7,500	15	112,500	
1GB for Ex000	7023A	2	7,125	60	427,500	
Dual SBus I/O daughter card	2730A	2	5,625	16	90,000	
System Service Proc	2751A	2	7,500	1	7,500	4,504
AC input module	3875A	2	2,250	4	9,000	
48 volt power supply	9685A	2	3,000	8	24,000	
Power Control Module	9681A	2	750	1	750	
Fan tray	9671A	2	1,500	16	24,000	
Control Board w/ Enet Cable	2720A	2	15,000	1	15,000	
Power Cord for System	3850A	2	0	4	0	
12 Meter SCSI cable	979A	2	199	2	398	
SBus Fast/Wide SCSI-2 (Diff)	1062A	2	971	2	1,943	
SBus Fibre Channel Host Adapter	6730A	2	2,025	42	85,050	
SBus Ultra Diff F/W SCSI	1065A	2	971	8	7,770	
Symmetra 16kva UPS	SX-SY16K	1	15,845	1	15,845	1,315
Subtotal					2,582,255	1,299,685
Storage						
StorEdge A3000 (15 X 9GB)	6534A	2	72,938	4	291,750	87,437
9.1 GByte Drive Module	X6516A	2	1,313	80	105,000	
Power Cord for A3000	3800A	2	0	4	0	
1200-GB Sun StorEdge A5200	SG-XARY523A-1200G	2	445,238	7	3,116,663	1,071,101
RSM Tray (7 X 9.1GB Disk)	6515AR4	2	13,950	2	27,900	
Subtotal					3,541,313	1,158,538
SunService Discount (10% Volume + 5% yearly prepayment)						(368,733)
Hardware Subtotal			371,011		6,123,568	2,089,489
Software						
Solaris 2.6 Media	SOLS-2.6MAR98	1	100	1	100	
S/W for SSP Version 3.1	SW-SSP-3.1	1	0	1	0	
SPARC Compiler, Single User Lic.	WCC-4.2-S	1	995	1	995	1,126
Sun StorEdge Vol Mgr. 2.6	VMR9S-260-R999	1	100	1	100	
RTU for E10000	VMR9S-260-S999	1	42,995	1	42,995	40,870
Oracle8i License (Version 8.1.5.2)		3	802,560	1	802,560	1,066,240
Subtotal					846,750	1,108,236
SunService Discount (10% Volume + 5% yearly prepayment)						(6,299)
Software Subtotal					846,750	1,101,937
Notes (Source):						
1. Sun Microsystems Inc.						
2. CAT Technology Inc						
3. Oracle Corp.						
			Total		6,970,318	3,191,426
Service for all hardware and software (except for Oracle			5Yr. cost		\$10,161,744	
Server) is from Sun Microsystems Inc.			QphD@1TB		35,878.1	
			\$/QphD@1TB		\$283	

Audited by: Francois Raab, Information Paradigm, Inc.

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

Note: Hardware and Sun Software Available Now; Oracle Software Available August 1, 1999.



Sun Starfire Enterprise 10000 using Oracle8i

TPC-D 1.3.1

Report Date:
February 12, 1999

Numerical Quantity Summary

Measurement Results:

Database Scaling Factor (SF/Size)	= 1,000 GB
Total Data Storage / Database Size	= 9.81
Database Load Time	= 46 hours 59 minutes
Query Streams for Throughput Test	= 0
TPC-D Power Metric (QppD@1000GB)	= 121,824.7
TPC-D Throughput Metric (QthD@1000GB)	= 10,566.3
Composite QphD@1000GB	= 35,878.1
Total System Price Over 5 Years	= \$10,161,744
TPC-D Price/Performance Metric	= \$283

Measurement Intervals:

Measurement Interval in Throughput Test (Ts)	= 5,792 seconds
Duration of Stream Execution:	

Stream ID	Seed	Start Date	Start Time	End Date	End Time
Stream00	1259440881	Tuesday, February 9	23:22:18 PST	Wednesday, February 10	00:58:50 PST
UF1		Tuesday, February 9	23:22:18 PST	Tuesday, February 9	23:55:00 PST
UF2		Wednesday, February 10	00:21:41 PST	Wednesday, February 10	00:58:50 PST

TPC-D Timing Intervals (in seconds)

Q1	Q2a	Q3	Q4	Q5	Q6	Q7	Q8b	Q9	Q10
0.2	57.5	89.2	70.3	0.7	0.2	0.8	14.2	593.1	200.5

Q11	Q12b	Q13	Q14c	Q15b	Q16	Q17b	UF1	UF2
56.0	54.9	5.4	0.5	140.8	274.6	42.2	1962.0	2229.2



Information Paradigm



Certified Auditor

Test Sponsors: Ray Glasstone
Director, Advanced Tech. Perf.
Oracle Corporation
100 Oracle Parkway
Redwood Shores, CA 94065

Brad Carlile
Manager, Strategic Apps. Eng.
Sun Microsystems, Inc.
8300 SW Creekside Place
Beaverton, OR 97008

February 12, 1999

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

Platform: Sun Starfire Enterprise 10000
DataBase Manager: Oracle8i Version 8.1.5.2
Operating System: Solaris Version 2.6
Other Software: Veritas Volume Manager Version 2.6

The results were:

CPU's Speed	Memory	Disks	QppD@1000GB	QthD@1000GB
Sun Starfire Enterprise 10000				
64 x UltraSPARC (400 MHz)	4 MB L2 each 60 GB main	1064 x 9.1 GB 14 x 9.1 GB (RSM)	121,824.7	10,566.3

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

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

-
- The database records were defined with the proper layout and size
 - The database was properly scaled to 1,000GB and populated accordingly
 - The database load time was correctly measured and reported
 - The ACID Properties were verified and met
 - The reported execution times were correctly measured and reported
 - Measurement repeatability was verified
 - At least 8 hours of database log was configured
 - The system pricing was verified for major components and maintenance
 - The major pages from the FDR were verified for accuracy

Additional Audit Notes:

None.

Respectfully Yours,

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 D Overview

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

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

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

- Give answers to real-world business questions
- Are far more complex than most OLTP transactions
- Include a rich breadth of operators and selectivity constraints
- Generate intensive activity on the part of the database server component of the system under test
- Are executed against a database complying to specific population and scaling requirements
- Are implemented with constraints derived from staying closely synchronized with an on-line production database

1. General Items

1.1 Benchmark Sponsor

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

Sun Microsystems and Oracle Corporation are the sponsors of this TPC-D 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*
- *Configuration parameters and options for any other software component incorporated into the pricing structure*
- *Compiler optimization options*

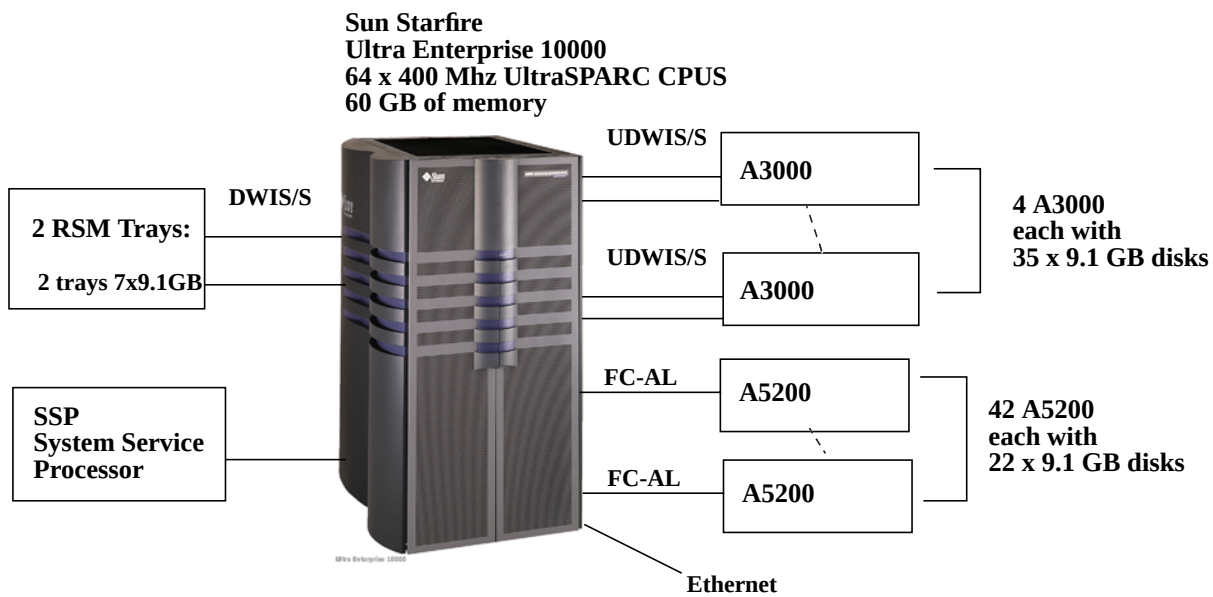
This requirement can be satisfied by providing a full list of all parameters and options.

Appendix A contains the Solaris and Oracle8i parameters used in this benchmark.

1.3 Configuration Diagram

Provide diagrams of both the measured and priced configurations, accompanied by a description of the differences. This includes, but is not limited to:

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



2. Clause 1 Logical Database Design

2.1 Database Definition Statements

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

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

2.2 Physical Organization

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

No record clustering or order clustering was used. Column ordering has been modified for some tables. Refer to the table create statements in Appendix B for further details.

2.3 Horizontal Partitioning

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

Horizontal partitioning was used for base tables LINEITEM, ORDERS, PARTS and CUSTOMER, for materialized view logs on LINEITEM and ORDERS and MJ_PLOCS, for materialized views MJ_PLOCS and MJ_PPSSRN and indexes L_ORED, IJ_L_ODATE, IJ_L_RDATE, IJ_L_PKEY, IJ_L_PTP, P_CBP, IJ_P_NNPS and IJ_P_NNPSPS. The details of this partitioning can be understood by examining the syntax of the table, materialized view and index definition statements in Appendix B. Similar partitioning was used in the qualification database, scaled appropriately for its size.

2.4 Replication

While there are some restrictions placed upon physical replication of objects in the test and qualification databases (see Clause 1.5.6), any such replication must be disclosed.

No replication of the base tables was used.

3. Clause 2 Queries and Update Functions

3.1 Query Language

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

SQL was the query language used to implement all queries.

3.2 Verifying Method for Random Number Generation

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

DBGEN version 1.3.1 and QGEN version 1.3.1 were used to verify random numbers for this TPC-D benchmark.

3.3 Generating Values for Substitution Parameters

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

The supplied QGEN version 1.3.1 was used to generate the substitution parameters.

3.4 Query Text and Output Data from Qualification Database

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

Appendix C contains the query text and query output.

3.5 Query Substitution Parameters and Seeds Used

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

Appendix D contains the seed and query substitution parameters.

3.6 Query Isolation Level

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

Level 1 isolation was used to run the queries.

3.7 Source Code of Update Functions

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

Multiple sqldr processes loaded data into temporary tables. Oracle8i's parallel insert and delete functionality was used to perform the Update Functions, selecting data from the temporary tables. See Appendix E.

3.8 Database Maintenance Option

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

This implementation uses the evolve option. Source code for this is included in Appendix E.

4. Clause 3 Database System Properties

4.1 Atomicity

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

4.1.1 Completed Transaction

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

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

4.1.2 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.

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

4.2 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 4.2.1.

1. The consistency of the ORDER and LINEITEM tables was verified based on a sample of O_ORDERKEY's.
2. 100 ACID Transactions were submitted from each of 2 execution streams.
3. The consistency of the ORDER and LINEITEM tables was reverified.

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.

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

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

1. An ACID Transaction, T1, was started for a randomly selected O_KEY, L_KEY, and DELTA. The ACID Transaction was suspended prior to COMMIT.

-
2. Another ACID Transaction, T2, was started using the same O_KEY and L_KEY and a randomly selected DELTA.
 3. T2 waited.
 4. T1 was allowed to COMMIT and T2 completed.
 5. It was verified that $T2.L_EXTENDEDPRICE = T1.L_EXTENDEDPRICE + (DELTA1 * (T1.L_EXTENDEDPRICE / T1.L_QUANTITY))$

4.3.4 Write-Write Conflict with Rollback

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

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

4.3.5 Concurrent Progress of Read and Write Transactions

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

1. An ACID Transaction, T1, was started for a randomly selected O_KEY, L_KEY, and DELTA. The ACID Transaction was suspended prior to COMMIT.
2. Another ACID Transaction, T2, was started which did the following:
For random values of PS_PARTKEY and PS_SUPPKEY, all columns of the PARTSUPP table for which PS_PARTKEY and PS_SUPPKEY are equal, are returned.
3. T2 completed.
4. T1 was allowed to COMMIT.
5. It was verified that appropriate rows in ORDER, LINEITEM and HISTORY tables were changed.

4.3.6 Read-Only Query Conflict with Update Transaction

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

-
1. A Transaction, T1, was started which executed Q17b (from Clause 2.19) against the qualification database, with p_brand = 'Brand#23' and p_container = 'MED BOX'.
 2. An ACID Transaction, T2, was started for a randomly selected O_KEY, L_KEY and DELTA.
 3. A Transaction, T3, was started which executed Q17b (from Clause 2.19) against the qualification database, with p_brand = 'Brand#23' and p_container = 'MED BOX'.
 4. T1 completed.
 5. T2 completed.
 6. T3 completed subsequently.
 7. It was verified that appropriate rows in ORDER, LINEITEM and HISTORY tables were changed.

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 3.5.2.

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-D database tables or recovery log tables.

The disks containing TPC-D tables were RAID-0, log files were mirrored. During the durability test one disk in the RAID-0 group for the data tables was removed from its cabinet, which caused the database to crash. The data files were restored from backup, and data recovery was done using the log file. The durability success file and the HISTORY table were compared and the counts matched.

Next the test was restarted and a disk containing one side of a log file mirror was removed. The test continued uninterrupted using the remaining side of the mirror for the log files.

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 reboot to recover.

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

4.4.3 Memory Failure

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

See the previous section.

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.

Table	Rows
Order	1,500,000,000
Lineitem	5,999,989,709
Customer	150,000,000
Part	200,000,000
Supplier	10,000,000
Partsupp	800,000,000
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.

Refer to Appendix F for details of the database layout.

5.3 Database partition/replication mapping

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

Horizontal partitioning was used for base tables LINEITEM, ORDERS, PARTS and CUSTOMER, for materialized view logs on LINEITEM and ORDERS, for materialized views MJ_LOCS and MJ_PPSSRN and indexes L_ORED, IJ_L_ODATE, IJ_L_RDATE, IJ_L_PKEY, IJ_L_PTP, P_CBP, IJ_P_NNPS and IJ_P_NNPS. The details of this partitioning can be understood by examining the syntax of the table, materialized view and index definition statements in Appendix B. Similar partitioning was used in the qualification database, scaled appropriately for its size.

Data is automatically replicated in the materialized views and transparently managed in a fully ACID fashion by the database manager. Only data in the base eight tables is referenced by the seventeen queries and insert and delete operations.

The database was not replicated in total.

5.4 RAID feature

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

Table/Index	RAID type
tables	RAID 0
indexes	RAID 0
materialized views	RAID 0
temp tablespace	RAID 0
log	RAID 1 + 0
usrtmp tablespace	RAID 0

5.5 Modifications to the DBGEN

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

The supplied DBGEN version 1.3.1 was used to generate the database population for this benchmark. The changes to DBGEN for partitioning of data are listed in Appendix B.

5.6 Database Content of Initial Ten Rows

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

Appendix G contains the first 10 rows of each table in the test database.

5.7 Database Load Time

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

The database load time was 46 hours 59 minutes.

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.

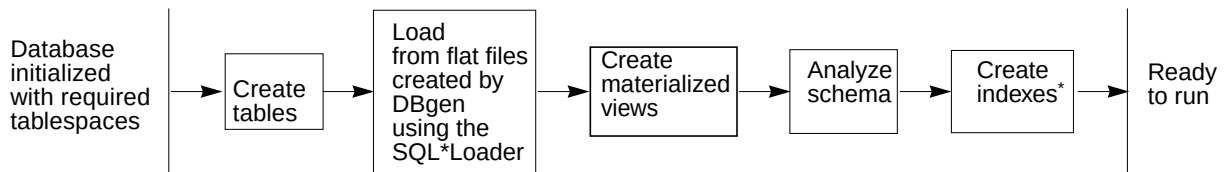
The data storage ratio is computed from the following information:

Disk Type	# of Disks	Space per Disk	Sub-Total Disk Space
A5200	924	9.1 GB	8408.4 GB
RSM Tray	14	9.1 GB	127.4 GB
A3000	140	9.1 GB	1274.0 GB
Total Space			9809.8 GB
Data Storage Ratio			9.81

5.9 Database Load Mechanism Details and Illustration

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

The database was loaded using data generation stored on flat files all on the tested and priced configurations. The SQL*Loader read the files that were created by the DBGEN program.



*Analyze index performed during index creation

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.

The following steps were used to implement the power test:

1. System boot
2. Database Restart
3. UF1 Update Transaction
4. Stream 00 Execution
5. UF2 Update Transaction

6.2 Timing Intervals for Each Query and Update Functions

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 are contained in the Numerical Quantity Summary in the Preface of this document.

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 only one execution stream. Since only one stream was used, the values for the power test are used for the throughput test.

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 are contained in the Numerical Quantity Summary in the Preface of this document.

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 5,792 seconds.

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

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

The start and finish time for each update function in the update stream are contained in the Numerical Quantity Summary in the Preface of this document.

6.7 Timing Intervals for Each Query and Each Update Function for Each Stream

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

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

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 in the Numerical Quantity Summary in the Preface of this document.

6.9 Reproducibility Method

Verify that the reported metrics are repeatable.

Performance results from the two executions of the TPC-D benchmark:

Run ID	QppD@1000GB	QthD@1000GB	QphD@1000GB
Run 1	124834.2	10769.0	36665.2
Run 2	121824.7	10566.3	35878.1

7. Clause 6 SUT and Driver Implementation

7.1 Driver

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

The Power Test is performed by a shell script called audit.sh. QGEN is first called with a stream id of 0 to generate the QET for the Power Test. UF1 is then started by executing the audit_uf1.sh script. Query submission follows, with the qexecpl.c ISL program. The execution of the UF2 script audit_uf2.sh rounds out the Power Test execution. Both wall-clock and high-resolution times are collected for all measurement intervals.

7.2 Implementation-Specific Layer

If any 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.

Query execution text generated by QGEN is picked up by the ISL program which submits the query to the SUT.

The ISL program (qexecpl.c) utilizes the Oracle Call Interface (OCI) to communicate with the Oracle database on the SUT. QETs directly generated by QGEN are read and submitted to the SUT via the ISL program (qexecpl.c) as dynamic SQL statements. The ISL program then fetches the query execution output and reports it to the user. Timings are taken at intervals specified by Clause 5.3.6.2.

7.3 Update Function

Multiple sqlldr processes loaded data into temporary tables. Oracle8i's parallel insert and delete functionality was used to perform the Update Functions, selecting data from the temporary tables.

8. Clause 7 Pricing

8.1 Hardware and Software Used

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

Refer to the Executive Summary.

8.2 Total Five Year Price

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

The total 5-year price of the configuration is \$10,161,744. For details of pricing, see the second page of the Executive Summary.

Hardware prices are from reseller price quotes. Refer to Appendix H.

8.3 Availability Date

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

All hardware components used in this benchmark and Solaris 2.6 are available now.

Oracle8i v 8.1.5.2 will be available August 1, 1999.

9. Auditor's Information and Attestation Letter

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

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

Appendix A. Programs and Scripts

Oracle8i Parameters

```
always_anti_join           = HASH
always_semi_join           = HASH
audit_trail                 = FALSE
compatible                 = 8.1.0
control_files               = /ora81/oracle813/980731/dbs/cntrl.dbf
#buffer_pool_keep          = (buffers:1500, lru_latches:1)
db_block_buffers            = 314280 #2.5G
db_block_checksum          = FALSE
db_block_lru_latches        = 160
db_block_max_dirty_target  = 0
db_block_size              = 8192
db_file_multiblock_read_count = 64
db_files                   = 1023
db_name                    = tpcd
db_writer_processes         = 10
distributed_transactions    = 0
dml_locks                  = 20000
enqueue_resources          = 64000
global_names               = FALSE
hash_area_size             = 40000000 #40m
#hash_area_size            = 30000000 #40m
hash_multiblock_io_count   = 64
large_pool_size            = 400000000 #400m
max_dump_file_size         = unlimited
max_rollback_segments      = 1024
nls_date_format            = YYYY-MM-DD
open_cursors               = 1024
optimizer_features_enable  = 8.1.0
optimizer_index_cost_adj   = 15
optimizer_mode              = CHOOSE
optimizer_percent_parallel = 100
optimizer_search_limit     = 6
parallel_broadcast_enabled = TRUE
parallel_execution_message_size = 4096
parallel_max_servers        = 2000
parallel_min_servers        = 640
processes                  = 2048
query_rewrite_enabled      = true
recovery_parallelism       = 64
replication_dependency_tracking = FALSE
sessions                   = 1024
shared_pool_size           = 250000000 #250m
sort_area_size             = 30000000 #30m
sort_area_retained_size    = 30000000 #30m
sql92_security             = FALSE
transaction_auditing        = FALSE
transactions               = 1024
transactions_per_rollback_segment = 1
query_rewrite_expression   = TRUE
log_checkpoints_to_alert   = true
parallel_adaptive_multi_user = true
```

Solaris Parameters

```
Wed Feb 10 12:05:23 PST 1999
*
* pageout tuning:
* Set lotsfree to 1/2 Gbyte, otherwise the default of 1/16 of all phys. mem.
* on our 64 Gbyte physical machine wastes several GB of memory
* minfree and desfree have been verified to be lotsfree/2 and lotsfree/4
* under 2.6
*
* DEFAULT: 1/16 physmem
* UNITS: pages (8kBytes)
*set lotsfree=0x10000
set lotsfree=0x20000
set priority_paging=1
* set cachefree to 30G to try to cut down on page/swap problems.
set cachefree=0x3C0000
*
*
* Boost time for autoup.
* This is the cycle time for a scan of all of memory. With large memory,
* the fsflush daemon will get rather busy running this scan using the default.
* DEFAULT: 30
* UNITS: Seconds
set autoup=600
*
* These come from the "starfire-specific" section in 2.5.1. They are not
* crucial, and definately exaggerate with the "DO NOT EDIT!" part.
* DO NOT EDIT!
set tune_t_gpgslow=250
set tune_t_minarmem=100
set tune_t_minasmem=250
set msgsys:msginfo_msgmap=200
set msgsys:msginfo_msgmni=100
set msgsys:msginfo_msgtql=80
set msgsys:msginfo_msgseg=2048
* 3.5 Gb
set shmsys:shminfo_shmmx=3758096384
* set shmsys:shminfo_shmmx=2042626048
* set shmsys:shminfo_shmmx=268435456
set shmsys:shminfo_shmsegs=200
set semsys:seminfo_semmmap=100
set semsys:seminfo_semmni=1000
set semsys:seminfo_semmns=4000
set semsys:seminfo_semmnu=800
```

```
set semsys:seminfo_semmnl=512
set semsys:seminfo_semtime=100
set pln:pln_enable_detach_suspend=1
set soc:soc_enable_detach_suspend=1
set socal:socal_enable_suspend=1
* SUNWecsr end
```

```
* set to allow bigger I/O's at least to test 10/6/97 by djm
* DEFAULT: 131072
* UNITS: Bytes
set maxphys=4194304
```

*

```
set moddebug=0x00001010
```

* set to 0 by rcl 9/26/97 it didn't help...

* DEFAULT: 6

* UNITS: selects case for these algorithms:

* 0 default for All sun4u (excluding starfire) uses various vaddr bits

* 1 physical address will be set to virtual address.

* 2 bin hopping

* 6 BSD page coloring scheme (Kessler's best bin) ** for starfire only **

*

```
set consistent_coloring=0
```

* added by pmb 7/11/97 Bump up FD hard limit

```
set rlim_fd_max=10240
```

* hope and a prayer

* set sq_max_draincnt=32

* BEGIN RAID Manager addition

* DO NOT EDIT from BEGIN above to END below...

forceload: drv/sd

forceload: drv/rdnexus

forceload: drv/rdriver

* END RAID Manager addition

* vxvm_START (do not remove)

forceload: drv/vxio

forceload: drv/vxspec

* the following 2 lines added 4/18/97 by djm as suggested by veritas

* the default is 256 but should never go above 65535

set vxio:vol_maxio=8192

set vxio:vol_maxkiocount=4096

* changing the following based on reading manual, djm 3/31/98

* memory for I/O's, default 4M

* set vxio:voliomem_max_memory=67108864

set vxio:voliomem_max_memory=134217728

* needs to be bigger than above, default 5M

* set vxio:voliomem_kvmap_size=73400320

set vxio:voliomem_kvmap_size=135266304

* memory for 1 I/O, default 512K

* set vxio:voliomem_base_memory=4194304

set vxio:voliomem_base_memory=104857600

* memory alloc chunk size, default 64K

set vxio:voliomem_chunk_size=1048576

* ioctl sizes, set as per A5000 experiments

set vxio:vol_maxioctl=131072

set vxio:vol_maxspecialio=10240

* vxvm_END (do not remove)

* Enable OBP Forth Debug --pboothby 6-dec-97

```
set forthdebug=1
```

* Set for new ISM-TSB stuff -- djm 4-mar-98

```
set enable_grp_ism=1
```

* Set for new shuffle stuff -- djm 24-mar-98

```
#set vb_shuffle=1
```

* Set ufs_WRITES to 0 to turn off ufs write throttles

```
set ufs:ufs_WRITES = 0
```

* This automatically makes cachefree 2 x lotsfree.

* set priority_paging = 1

* Set kmem_flags for ISM mode debug - By Dr. E. on 01/26/99

* how to set live kernel variables

* adb -kw

* kmem_flags/W 0xe

* kmem_debug_enable/W -1

*

* comment out to boot this way:

* set kmem_flags = 0xe

* set kmem_debug_enable = -1

Wed Feb 10 12:05:24 PST 1999

Appendix B. Programs and Scripts

ACID Source

atrans.sql

```
rem =====+
rem      Copyright (c) 1995 Oracle Corp, Redwood Shores, CA      |
rem      OPEN SYSTEMS PERFORMANCE GROUP                          |
rem      All Rights Reserved                                      |
rem =====+
rem      FILENAME
rem      atrans.sql
rem      DESCRIPTION
rem      Creates ACID Transaction Package for TPC-D benchmark.
rem      Asks user to input values for o_key, delta and output file.
rem      Update: 10/24/1998      Barry N. Perkins
rem      Added commit inside loop so that the
rem      exception handlers can handle the 8177
rem      on commit. Added exception handler for
rem      604, error in recursive SQL to handle
rem      8177 in recursive sql.
rem =====+

set serverout on;
set termout on;
set echo on;

CREATE OR REPLACE PACKAGE d_atrans
IS
PROCEDURE doatrans
(
  l_keyIN OUT integer,
  o_keyIN OUT integer,
  deltaIN OUT integer,
  l_pkey      IN OUT integer,
  l_skey      IN OUT integer,
  l_guan      IN OUT integer,
  l_newquan   IN OUT integer,
  l_tax       IN OUT number,
  l_disc      IN OUT number,
  l_eprice    IN OUT number,
  l_newprice  IN OUT number,
  o_tpriceIN OUT number,
  o_newtpriceIN OUT number,
  rpriceIN OUT number,
  costIN OUT number
);
END;
/

CREATE OR REPLACE PACKAGE BODY d_atrans
IS
PROCEDURE doatrans
(
  l_keyIN OUT integer,
  o_keyIN OUT integer,
  delta IN OUT integer,
  l_pkey      IN OUT integer,
  l_skey      IN OUT integer,
  l_guan      IN OUT integer,
  l_newquan   IN OUT integer,
  l_tax       IN OUT number,
  l_disc      IN OUT number,
  l_eprice    IN OUT number,
  l_newprice  IN OUT number,
  o_tpriceIN OUT number,
  o_newtpriceIN OUT number,
  rpriceIN OUT number,
  costIN OUT number
)
IS
  ottotal number;
  not_serializable EXCEPTION;
  not_serializable_recurs EXCEPTION;
  PRAGMA EXCEPTION_INIT(not_serializable,-8177);
  PRAGMA EXCEPTION_INIT(not_serializable_recurs,-604);
BEGIN
  LOOP BEGIN

    select o_totalprice
    into o_tprice
    from orders
    where o_orderkey = o_key;

    select l_quantity, l_extendedprice, l_partkey, l_suppkey, l_tax, l_discount
    into l_guan, l_eprice, l_pkey, l_skey, l_tax, l_disc
    from lineitem
    where l_orderkey = o_key
    and l_linenumber = l_key;

    ottotal := o_tprice - trunc((trunc((l_eprice * (1.0-l_disc)),2) *
    (1.0+l_tax)),2);
    rprice := trunc((l_eprice/l_guan), 2);
    cost := trunc((rprice * delta), 2);
    l_newprice := l_eprice + cost;
    o_newtprice := trunc((l_newprice * (1.0 - l_disc)), 2);
    o_newtprice := ottotal + trunc((o_newtprice * (1.0 + l_tax)), 2);
    l_newquan := l_guan + delta;

    update lineitem
    set l_extendedprice = l_newprice,
```

```
l_quantity = l_newquan
where l_orderkey = o_key
and l_linenumber = l_key;

update orders
set o_totalprice = o_newtprice
where o_orderkey = o_key;

insert into history (h_p_key, h_s_key, h_o_key, h_l_key, h_delta, h_date_t)
values (l_pkey, l_skey, o_key, l_key, delta, sysdate);

EXIT;

EXCEPTION
WHEN not_serializable THEN
  ROLLBACK;
WHEN not_serializable_recurs THEN
  ROLLBACK;
END;

END LOOP;

END doatrans;
END;
/

exit;
```

a_query.sql

```
rem =====+
rem      Copyright (c) 1996 Oracle Corp, Redwood Shores, CA      |
rem      OPEN SYSTEMS PERFORMANCE GROUP                          |
rem      All Rights Reserved                                      |
rem =====+
rem      FILENAME
rem      a_query.sql
rem      DESCRIPTION
rem      Performs ACID Query for TPC-D benchmark.
rem      Asks user to input values for o_key
rem =====+
rem Usage:      sqlplus tpcd/tpcd @a_query <o_key>
rem

set serverout on;

select
'BEFORE ACID QUERY' as STAGE,
substr(TO_CHAR(sysdate,'YYYY-MM-DD HH:MI:SS'),1,20) as CURRENT_TIME
from dual;

select SUM(trunc(trunc(l_extendedprice * (1-l_discount),2) * (1+l_tax),2)) AS
RESULT
from lineitem
where l_orderkey = &l;

select
'AFTER ACID QUERY' as STAGE,
substr(TO_CHAR(sysdate,'YYYY-MM-DD HH:MI:SS'),1,20) as CURRENT_TIME
from dual;

exit;
```

a_q17b.sql

```
set serverout on;

select
'BEFORE Q17B' as STAGE,
substr(TO_CHAR(sysdate,'YYYY-MM-DD HH:MI:SS'),1,20) as CURRENT_TIME
from dual;

select /*+ NOPARALLEL(LINEITEM) */
sum(l_extendedprice) / 7.0 as avg_yearly
from
  lineitem,
  (select /*+ NOPARALLEL(PARTS) NOPARALLEL(LINEITEM) */
    p_partkey,
    0.2 * avg(l_quantity) as avgqty
  from parts, lineitem
  where
    p_partkey = l_partkey and
    p_brand = 'Brand#23' and
    p_container = 'MED BOX'
  group by p_partkey
  ) avg_quantity
where
  l_partkey = p_partkey and
  l_quantity < avgqty;

select
'AFTER Q17B' as STAGE,
substr(TO_CHAR(sysdate,'YYYY-MM-DD HH:MI:SS'),1,20) as CURRENT_TIME
from dual;

exit;
```

Load Process

bumpx.pl

```
#!/opt/GOODies/bin/perl
#
# $Header: bumpx.pl 17-sep-98.16:37:34 mpoess Exp $
#
# bumpx.pl
#
# Copyright (c) Oracle Corporation 1998. All Rights Reserved.
#
#
#      MODIFIED      (MM/DD/YY)
#      mpoess        09/17/98 - porting to nt
#      mpoess        08/04/98 - add hash/composite partitioning
#      akarasik      07/29/98 -
#      akarasik      07/29/98 - Checking in
#
#!/usr/local/bin/perl
#
# $Header: bumpx.pl 17-sep-98.16:37:34 mpoess Exp $
#
#      NAME
#      bumpx.pl - BenchMark Package eXpert
#      DESCRIPTION
#      Utility for working with benchmark databases
#      RETURNS
#
#      NOTES
#
#      MODIFIED      (MM/DD/YY)
#      kwong 05/22/98 - add change storage for tables and indexes in
#                  chdop
#      kwong 05/21/98 - fix load pipe
#      kwong 05/21/98 - add some more default values in init.ora
#      kwong 05/20/98 - add parameter nt_port
#      kwong 05/19/98 - fix control file generation
#      kwong 05/18/98 - run utlxplan.sql after created user
#      kwong 05/18/98 - add "dbgen" phase
#      kwong 05/15/98 - calculate partition values for l_orderkey
#                  and l_partkey
#      kwong 05/13/98 - add phase "chdop" for modifying dop as the
#                  end of the build
#      kwong 05/13/98 - fix the order of create tablespaces and add
#                  datafiles
#      pswong 04/15/97 - fix pipeline load
#      pswong 04/02/97 - analyze partition support
#      pswong 03/27/97 - introduced ts groups
#      pswong 03/18/97 - named pipe loader and dbgen support
#      pswong 02/28/97 - more partition support
#      pswong 02/14/97 - Version 8
#      amozes 10/27/94 - Creation
#
#*****
# Main Section
#*****
# changes:
# 091598: MPOESS - added support for hints in the as select clause
#
$os = $ENV{'OS'};
if (($os cmp 'Windows_NT') != 0)
{# os is UNIX
    $os = "unix";
}
else
{
    $os = "nt";
}

$| = 1;
$verbose = 0;
$allphases = "dbcre,shutd,start,dbgen,scrcr,scuto,scuvo,sectso,dapop,ixcre,analyz,chob,expln,quer
y,sdgen,plcre";
if (($os cmp "unix")==0)
{
    $defphases = "shutd,start,dbcre,sectso,scuto,dapop,scuvo,analyz,ixcre,chob";
}
else
{
    $defphases =
"sdgen,shutd,start,dbgen,plcre,dbcre,sectso,scuto,dapop,scuvo,analyz,ixcre,chob";
}
$allbmtypes = "tpcd,wisc";
$bmttype = "tpcd" if !defined $bmttype;
$pdfile = "param.txt"; # This file contains the description of all possible
parameters.
&read_parameter_description;
$runsilent=0;
while ($arg = shift(@ARGV))
{
    if ($arg != -/i|o|t|c|x|p|d|a|s/)
    {
        $error = "*** Error: Bad argument to $0: $arg\n";
        &usage;
    }
    $runsilent = 1 if ($arg == -/s/);
    $infile = shift(@ARGV) if ($arg == -/i/);
    $outfile = shift(@ARGV) if ($arg == -/o/);
    $bmttype = shift(@ARGV) if ($arg == -/t/);
    $dcreate = 1 if (($arg == -/c/) || ($arg == -/xc/));
    $doexecute = 1 if (($arg == -/x/) || ($arg == -/cx/));
    $phasetest = shift(@ARGV) if ($arg == -/p/);
    if ($arg == -/d/)
    {
        $defpar = shift(@ARGV);
    }
}
```

```
&defaults;
@keys = keys %params;
while ($#keys >=0)
{
    $key = pop(@keys);
    if (($defpar cmp "") ==0)
    {
        print $key, "=", $params{$key}, "\n";
    }
    else
    {
        print $key, "=", $params{$key}, "\n" if ($key != -/defpar/);
    }
}
exit(0);
}
if ($arg == -/a/)
{
    $infile = "bumpx.conf" if !defined $infile;
    $filedir = $ENV{'BUMPX_DIR'};
    $infile = $filedir.$infile;
    &readfile;
    &defaults;
    @keys = keys %params;
    while ($#keys >=0)
    {
        $key = pop(@keys);
        print $key, "=", $params{$key}, "\n";
    }
}
exit(0);
}
}

$dcreate = 0 if !defined $dcreate;
$doexecute = 0 if !defined $doexecute;
if (!($dcreate || $doexecute))
{
    $error = "*** Error: Must specify either -c or -x or both\n";
    &usage;
}
$infile = "bumpx.conf" if !defined $infile;
$outfile = "bumpx.dat" if !defined $outfile;
$filedir = $ENV{'BUMPX_DIR'};
$infile = $filedir.$infile;
$outfile = $filedir.$outfile;

if (($os cmp "nt") == 0)
{
    ## NT Port (Use tmpfile to buffer commands and nrntpb to synchronize
    them)
    $tmpfile = "tmp.txt";
    $tmpfile = $filedir.$tmpfile;
    $nrntpb = "nrntpb.exe";
    ## NT End
}

if ((!-e $infile) && !$doexecute && $dcreate)
{
    $error = "*** Error: -i file, $infile, does not exist\n";
    &usage;
}
if ((!-e $outfile) && !$dcreate)
{
    $error = "*** Error: -o file, $outfile, does not exist\n";
    &usage;
}
$phasetest = $defphases if !defined $phasetest;
@phases = split(/,/,$phasetest);

open (OUTFILE, ">$outfile") if $dcreate;
&readfile;
&defaults;
&dcreate if $dcreate;
close (OUTFILE) if $dcreate;

## NT Port (Use tmpfile to buffer commands for nrntpb)
open (TMPFILE, ">$tmpfile") if ( (($os cmp "nt") == 0) && $doexecute);
## NT End

&doexecute if $doexecute;

## NT Port
close(TMPFILE) if ( (($os cmp "nt") == 0) && $doexecute);
## NT End

exit(0);

sub readfile
{
    $matchon = 0;
    $contin = 0;
    $pkey = "";
    $pval = "";
    open (INFILE, "$infile");
    WLOOP:
    while ($line = <INFILE>)
    {
        $line = $line."\\n" if $line != -/\\n/;
        study $line;
        if ($line =~ /^\\*matchon/)
        {
            $matchon = 1;
            next WLOOP;
        }
        if ($line =~ /^\\*matchoff/)
        {
            $matchon = 0;
            next WLOOP;
        }
        if ($matchon == 1)
        {
            $pkey = $line;
            $pval = <INFILE>;
            $line = <INFILE>;
        }
    }
}
```



```

&dump0("#####");
&dump0("# Database Creation Phase");
&time0("Begin database creation");

# Create init.ora files
&createios(); # (This will be done before eval)

# Shut down anything that may have been running
&shutdb(); # (This will be done before eval)

# Create database (system tablespace/datafile, redo logs, rollback segments)
@sys_files = split (/,/, $params{'ts_sys_datafiles'});
$sys_file = shift(@sys_files);
@log_files = split (/,/, $params{'ts_log_datafiles'});
&dump0("# creating database and initial rollback segment");
$cmd .= "startup pfile=$iofname[1] nomount;\n"; # ^^su
$cmd .= "create database\n";
$cmd .= " controlfile reuse\n" if !($params{'db_ctlreuse'} =~ /false/);
for ($i = 0; $i < $params{'ts_log_files_pt'}; $i++)
{
    ($i == 0) ? ($addname = "logfile") : ($addname = " ");
    (($i+1) == $params{'ts_log_files_pt'}) ? ($addcomma = "") : ($addcomma = ",");
    $log_file = shift(@log_files);
    if (($os cmp "unix") == 0)
    {
        $cmd .= " $addname '$params{'ts_log_area'}$log_file' size $params{'ts_log_size'} $params{'ts_log_options'} $addcomma\n";
    }
    else
    {
        $cmd .= " $addname (' $params{'ts_log_area'}$log_file') size $params{'ts_log_size'} $params{'ts_log_options'} $addcomma\n";
    }
}
$cmd .= " maxlogfiles $params{'db_maxlogfiles'}\n" if defined $params{'db_maxlogfiles'};
$cmd .= " maxlogmembers $params{'db_maxlogmembers'}\n" if defined $params{'db_maxlogmembers'};
$cmd .= " maxloghistory $params{'db_maxloghistory'}\n" if defined $params{'db_maxloghistory'};
$cmd .= " datafile '$params{'ts_sys_area'}$sys_file' size $params{'ts_sys_size'} $params{'ts_sys_options'}\n";
$cmd .= " maxdatafiles $params{'db_maxdatafiles'}\n" if defined $params{'db_maxdatafiles'};
$cmd .= " maxinstances $params{'db_maxinstances'}\n" if defined $params{'db_maxinstances'};
$cmd .= " maxarchlogs $params{'db_maxarchlogs'}\n" if defined $params{'db_maxarchlogs'};
$cmd .= " archivelog\n" if defined $params{'db_archivelog'};
$cmd .= " character set $params{'db_charset'}\n" if defined $params{'db_charset'};
$cmd .= "\n";
$cmd .= "\n";
$cmd .= "create public rollback segment t_rsl ";
$cmd .= " storage $params{'ts_undo_rs_storage'}" if defined $params{'ts_undo_rs_storage'};
$cmd .= "\n";
$cmd .= "\n";
$cmd .= "alter rollback segment t_rsl online;\n";
$cmd .= "\n";
$cmd .= "shutdown\n"; # ^^sd
&dump1("sql");
&dump0("wait");

# This startup is in its own session because of some weird bug I've been
# seeing on the SP2; otherwise, it's in the previous session
# ^^su
$cmd = "";
$cmd .= "startup pfile=$iofname[1];\n";
&dump1("sql");
&dump0("wait");

if ($params{'skip_ts'} != /undo/)
{
    @ts_datafiles = split(/,/, $params{'ts_undo_datafiles'});
    $ts_datafile = shift(@ts_datafiles);
    $cmd .= "create tablespace ts_undo\n";
    $cmd .= " datafile '$params{'ts_undo_area'}$ts_datafile' size $params{'ts_undo_first_size'} $params{'ts_undo_options'}\n";
    &dump1("sql");
    &dump0("wait");
    &add_dfs("ts_undo");
}

# currently set up for the foreground - maybe should be changed
&dump0("# creating extra rollback segments");
for ($i = 1; $i <= $params{'ts_undo_rs'}; $i++)
{
    $cmd .= "create public rollback segment r$i";
    $cmd .= " storage $params{'ts_undo_rs_storage'}" if defined $params{'ts_undo_rs_storage'};
    $cmd .= " tablespace ts_undo" if ($params{'skip_ts'} != /undo/);
    $cmd .= "\n";
}
&dump1("sql");
&dump0("wait");

# currently set up for the foreground - maybe should be changed
&dump0("# creating extra logfile threads");
for ($i = 1; $i < $params{'ts_log_threads'}; $i++)
{
    $thrno = $i + 1;
    $cmd .= "alter database add logfile thread $thrno\n";
    for ($j = 0; $j < $params{'ts_log_files_pt'}; $j++)
    {
        ($addcomma = (($j+1) == $params{'ts_log_files_pt'}) ? ($addcomma = ",") : "");
        $log_file = shift(@log_files);
        $cmd .= " '$params{'ts_log_area'}$log_file' size $params{'ts_log_size'} $params{'ts_log_options'} $addcomma\n";
    }
    $cmd .= "alter database enable public thread $thrno;\n";
}
&dump1("sql");

&dump0("wait");

# Build data dictionary
&dump0("# building data dictionary");
$cmd .= "set termout off\n";
$cmd .= "set echo off\n";
$cmd .= '@' . $params{'dd_sql_area'} . " . catalog.sql;\n";
$cmd .= '@' . $params{'dd_sql_area'} . " . catparr.sql;\n";
$cmd .= '@' . $params{'dd_sql_area'} . " . catproc.sql;\n";
# $cmd .= '@' . $params{'dd_sql_area'} . " . utlxplan.sql;\n";
if (($os cmp "unix") != 0)
{
    # nt
    $cmd .= "connect system/manager\n";
    $cmd .= '@' . $params{'dd_dbs_area'} . " . pupbld.sql;\n";
}
&dump1("sql");
&dump0("wait");
&time0("Done database creation");

# prepare for multi-user
# ^^sd
$cmd .= "shutdown;\n";
&dump1("sql");
&dump0("wait");

# ^^su
&startdb();
}# end dbcre

sub shutd
{
    &dump0("#####");
    &dump0("# Shutdown Database - All Instances");
    &shutdb();
    &time0("Done database shutdown - all instances");
}

sub start
{
    &dump0("#####");
    &dump0("# Startup Database - All Instances");
    &startdb();
    &time0("Done database startup - all instance");
}

sub sdgen
{
    &dump0("#####");
    &dump0("# Data (Seed File) Generation Phase");
    &time0("Begin creating seed files");

    $params{'load_tables'} = $params{'tab_tables'} if !defined $params{'load_tables'};
    @load_tablelist = split(/,/, $params{'load_tables'});
    foreach $table (@load_tablelist)
    {
        $dbgopt = "";
        $dbgprc = ((defined $params{'load_dbgen_'.$table.'_option_C'}) ? $params{'load_dbgen_'.$table.'_option_C'} : $params{'load_dbgen_def_option_C'});

        if (! $seen{$dbgprc}++) {
            $dbgopt = $verbose ? "-v " : "";
            $dbgopt .= "-O s -s " . $params{'scale_factor'};
            $cmd .= $dbgopt . " -C " . $dbgprc;
            $cmd .= sprintf(" 2>> dbgen_seed_%d.log", $dbgprc) if $log_output;
            $cmd .= "\n";
            &dump1("dbgen");
            &time0("Done creating seed files for degree $dbgprc");
        }
        &dump0("wait");
        &time0("Done creating seed files");
        &dump0("wait");
    } # end of sdgen
}

sub dbgen
{
    &dump0("#####");
    &dump0("# Data (Flat File) Generation Phase");
    &time0("Begin creating flat files");

    $scur_inst = 1 if $multi; # for possible locality on SP2
    ($params{'load_type'} =~ /delim/) ? ($ud = 1) : ($ud = 0);
    $params{'load_tables'} = $params{'tab_tables'} if !defined $params{'load_tables'};
    @load_tablelist = split(/,/, $params{'load_tables'});
    foreach $table (@load_tablelist)
    {
        $scurs = $params{'tab_'.$table.'_ts'};
        $sdp = (defined $params{'tab_'.$table.'_load_degpar'}) ? $params{'tab_'.$table.'_load_degpar'} : $params{'load_deg_parallel'};
        $usels = 0;
        $lfexist = 0;
        $bfexist = 0;
        $dafexist = 0;
        $difexist = 0;
        $ffexist = 0;

        # see if we are using pipes
        $up = $params{'load_use_pipes'} =~ /$table/ ? 1 : 0;

        $dgp = ($params{'load_dbgen_partition'} =~ /$table/ ? 1 : 0;
        $dbgpar = (defined $params{'load_dbgen_'.$table.'_option_C'}) ?

```

```

$parms{'load_dbgen_'.$table.'_option_C'} :
$parms{'load_dbgen_def_option_C'};

# create all of the "executable" load sections
# These variables will hold for all loaders for a given table
$upwd = "";
$parbool = "";
$dbrbool = "";
$silent = "";
$dismax = "";

$file = "";
$errors = "";
$rows = "";
$bsize = "";
$load = "";
$skip = "";

$dbgopt = "";
$dbgtab = "";

if ($table =~ /lineitem/)
{
    $dbgtab = "L";
}
elseif ($table =~ /orders/)
{
    $dbgtab = "O";
}
elseif ($table =~ /partsupp/)
{
    $dbgtab = "S";
}
elseif ($table =~ /parts/)
{
    $dbgtab = "P";
}
elseif ($table =~ /customer/)
{
    $dbgtab = "C";
}
elseif ($table =~ /supplier/)
{
    $dbgtab = "S";
}
elseif ($table =~ /nation/)
{
    $dbgtab = "N";
}
elseif ($table =~ /region/)
{
    $dbgtab = "R";
}

$dbgopt .= "-T ".$dbgtab." ";
$dbgopt .= "-s ".$parms{'scale_factor'} . " ";
$dbgopt .= "-C ";
$dbgopt .= ((defined $parms{'load_dbgen_'.$table.'_option_C'}) ?
    $parms{'load_dbgen_'.$table.'_option_C'} :
    $parms{'load_dbgen_def_option_C'});
if ($dgp)
{
    $dbgopt .= "-p -i ";
    $dbgopt .= ((defined $parms{'load_dbgen_'.$table.'_input_params'}) ?
    $parms{'load_dbgen_'.$table.'_input_params'} :
    $parms{'load_dbgen_def_input_params'});
}

# Start - Barry N. Perkins 8/27/1998
# included the following lines to copy flat files#
# from default location to backup location
# Defined a new variable for backup location
$fbkp = ((defined $parms{'load_dbgen_flatfile_backup'}) ?
    $parms{'load_dbgen_flatfile_backup'} :
    0);
$bpre = ((defined $parms{'load_dbgen_backup_'.$table.'_prefix'}) ?
    $parms{'load_dbgen_backup_'.$table.'_prefix'} :
    $parms{'load_dbgen_backup_area'} . $table);
# End - Barry N. Perkins 8/27/1998
# if using pipes, create the pipes

if ($dgp)
{
    # create script to generate partitioned flat files for each dbgpar

    $nperset = 1;

    if ($dgp && $dp > 1)
    {
        $nset = ($parms{'tab_'.$table.'_part'} > $dp) ? 1 : int($dp /
        $parms{'tab_'.$table.'_part'});
        $nperset = int($dbgpar / $nset);
    }

    for ($i=0;$i<$dbgpar;$i++)
    {
        $cmd = sprintf("%s -S %d", $dbgopt, ($i + 1));
        if ($dgp)
        {
            # if the output prefix name is specified with #
            # divide output file names into $dp/#numpart sets
            $fpre = ((defined $parms{'load_dbgen_'.$table.'_output_prefix'}) ?
            $parms{'load_dbgen_'.$table.'_output_prefix'} :
            $parms{'load_flatfile_area'} . $table);
            if ($fpre =~ /\#/)
            {
                $snum = 1;
                $snum = (int($i/$nperset) + 1) if ($dp > 1);
                $fpre =~ s/\#/$snum/g;
            }
            $cmd .= sprintf("-o %s_%d ", $fpre, ($i + 1));
        }
        $cmd = $verbose ? "-v -f " . $cmd : "-f " . $cmd;
        $cmd .= sprintf(" 2>> %s_%d.log", $table, ($i + 1)) if $log_output;

#cmd .= " & ";
$cmd .= "\n";
&dump1("dbgen");
}
&dump0("wait");
else
{
    $cmd = $verbose ? "-v " : "";
    $cmd .= "-f " . $dbgopt;
    $cmd .= sprintf(" 2>> %s.log", $table) if $verbose;
    $cmd .= "\n";
    &dump1("dbgen");
}
}

# Start - Barry N. Perkins 8/27/1998
# Included the following lines to copy flat files#
# from default location to backup location
# Copy from default location to backup location
if ($fbkp)
{
    $cmd = sprintf("copy %s%s.* %s.*\n", $fpre, $table, $bpre);
    &dump1("sh");
    &dump0("wait");
}

# End - Barry N. Perkins 8/27/1998
&dump0("wait");
&time0("Done creating flat files for table $table");
&time0("Done creating flat files");
} # end of dbgen

sub dbgen_my_split_version # I will delete this and dbgen_nt and dbgen_unix as
soon as new dbgen is fully tested
{
    if (($os cmp "unix") == 0)
    {
        &dbgen_unix();
    }
    else
    {
        &dbgen_nt();
    }
}

sub dbgen_nt
{
    &dump0("#####");
    &dump0("# Data (Flat File) Generation Phase");
    &time0("Begin creating flat files");

    $cur_inst = 1 if $multi; # for possible locality on SP2
    ($parms{'load_type'} =~ /delim/) ? ($ud = 1) : ($ud = 0);
    $parms{'load_tables'} = $parms{'tab_tables'} if !defined
    $parms{'load_tables'};
    @load_tablelist = split(/,/, $parms{'load_tables'});
    foreach $table (@load_tablelist)
    {
        $curs = $parms{'tab_'.$table.'_ts'};
        $dp = $parms{'tab_'.$table.'_load_degpar'};
        $usels = 0;
        $lfexist = 0;
        $bfexist = 0;
        $dafexist = 0;
        $difexist = 0;
        $ffexist = 0;

        $up = 0;
        $dgp = 0;
        $dbgpar = 1;

        # see if we are using pipes

        $up = 1 if ($parms{'load_use_pipes'} =~ /$table/);
        $dgp = 1 if ($parms{'load_dbgen_partition'} =~ /$table/);
        $dbgpar = $parms{'load_dbgen_def_option_C'} if defined
        $parms{'load_dbgen_def_option_C'};
        $dbgpar = $parms{'load_dbgen_'.$table.'_option_C'} if defined
        $parms{'load_dbgen_'.$table.'_option_C'};

        # create all of the "executable" load sections
        # These variables will hold for all loaders for a given table
        $upwd = "";
        $parbool = "";
        $dbrbool = "";
        $silent = "";
        $dismax = "";

        $file = "";
        $errors = "";
        $rows = "";
        $bsize = "";
        $load = "";
        $skip = "";

        $dbgopt = "";
        $dbgtab = "";

        if ($table =~ /lineitem/)
        {
            $dbgtab = "L";
        }
        elseif ($table =~ /orders/)
        {
            $dbgtab = "O";
        }
        elseif ($table =~ /partsupp/)
        {
            $dbgtab = "S";
        }
        elseif ($table =~ /parts/)
        {
            $dbgtab = "P";
        }
    }
}

```

```

    }
    elif ($table == /customer/)
    {
$dbgtab = "c";
    }
    elif ($table == /supplier/)
    {
$dbgtab = "s";
    }
    elif ($table == /nation/)
    {
$dbgtab = "n";
    }
    elif ($table == /region/)
    {
$dbgtab = "r";
    }

if ($up) {
    $dbgopt = "-T ".$dbgtab." ";
    $dbgopt = "-s " . $params{'scale_factor'} . " ";
    $dbgopt = "-C ";
    $dbgopt = ((defined $params{'load_dbgen_'.$table.'_option_C'}) ?
$params{'load_dbgen_'.$table.'_option_C'} :
$params{'load_dbgen_def_option_C'});
    if ($dgp) {
        $dbgopt = "-p -i ";
        $dbgopt = ((defined $params{'load_dbgen_'.$table.'_input_params'}) ?
$params{'load_dbgen_'.$table.'_input_params'} :
$params{'load_dbgen_def_input_params'});
    }
} else
{
    $dbgopt = "-T ".$dbgtab." ";
    $dbgopt = "-s " . $params{'scale_factor'} . " ";
    $dbgopt = "-C ";
    $dbgopt = ((defined $params{'load_dbgen_'.$table.'_option_C'}) ?
$params{'load_dbgen_'.$table.'_option_C'} :
$params{'load_dbgen_def_option_C'});
}

# if using pipes, create the pipes

if($up)
{
    # create script to generate partitioned flat files for each dbgp

    $nperset = 1;

    if ($dgp && $dp > 1) {
        $nset = int($dp / $params{'tab_'.$table.'_part'});
        $nperset = int($dbgp / $nset);
    }

    for ($i=0;$i<$dbgp;$i++) {
        $cmd = sprintf("%s -S %d", $dbgopt, ($i + 1));
        if ($dgp) {
            # if the output prefix name is specified with #
            # divide output file names into $dp/numpart sets
            $fpre = (defined $params{'load_dbgen_'.$table.'_output_prefix'}) ?
$params{'load_dbgen_'.$table.'_output_prefix'} :
$params{'load_dbgen_def_output_prefix'};
            if ($fpre == /\#/) {
                $snnum = 1;
                $snnum = (int($i/$nperset) + 1) if ($dp > 1);
                $fpre = s/\#/$snnum/g;
            }
            $cmd = sprintf("-o %s %d", $fpre, ($i + 1));
        }

        $cmd = "-f " . $cmd;
        #cmd = " & ";

        $cmd = "\n";
        &dumpl("dbgen");
    }

    &dump0("wait");

    # create script to concatenate partitioned flat files for each dbgp
    into partitioned flat files by partitions and number of partitioned files in each
    partition

    if ($dgp) {
        $nptn = $params{'tab_'.$table.'_part'};
        for ($i=0;$i<$dbgp;$i++) {
            # if the output prefix name is specified with #
            # divide output file names into $dp/numpart sets
            $fpre = (defined $params{'load_dbgen_'.$table.'_output_prefix'}) ?
$params{'load_dbgen_'.$table.'_output_prefix'} :
$params{'load_dbgen_def_output_prefix'};
            if ($fpre == /\#/) {
                $snnum = 1;
                $snnum = (int($i/$nperset) + 1) if ($dp > 1);
                $fpre = s/\#/$snnum/g;
            }
            $pnnum = ($i % $nptn) + 1;
            $cmd = sprintf("copy %s_*.d %s %d\n", $fpre, $pnnum, $fpre, $pnnum);
            &dumpl("sh");
            &dump0("wait");
            $cmd = sprintf("del %s_*.d\n", $fpre, $pnnum);
            &dumpl("sh");
            &dump0("wait");
        }
    }
} else
{
    $cmd = "-f " . $dbgopt;
    $cmd = "\n";
    &dumpl("dbgen");
}

&dump0("wait");

&time0("Done creating flat files for table $table");
}
&dumpl0("wait");
&time0("Done creating flat files");
} # dbgen_nt

sub dbgen_unix
{
    &dumpl0("#####");
    &dumpl0("# Database Population Phase");

    $cur_inst = 1 if $multi; # for possible locality on SP2
    ($params{'load_type'} == /delim/) ? ($ud = 1) : ($ud = 0);
    $params{'load_tables'} = $params{'tab_tables'} if !defined
$params{'load_tables'};
    @load_tablelist = split(/,/, $params{'load_tables'});
    foreach $table (@load_tablelist)
    {
        $scurts = $params{'tab_'.$table.'_ts'};
        $sdp = $params{'tab_'.$table.'_load_degpar'};
        $susels = 0;
        $lfexist = 0;
        $bfexist = 0;
        $dafexist = 0;
        $difexist = 0;
        $ffexist = 0;

        $up = 0;
        $dgp = 0;
        $dbgp = 1;

        # see if we are using pipes

        $up = 1 if ($params{'load_use_pipes'} == /$table/);
        # <bug> parts/partsupp can be confused
        $dgp = 1 if ($params{'load_dbgen_partition'} == /$table/);
        $dbgp = $params{'load_dbgen_def_option_C'} if defined
$params{'load_dbgen_def_option_C'};
        $dbgp = $params{'load_dbgen_'.$table.'_option_C'} if defined
$params{'load_dbgen_'.$table.'_option_C'};

        $params{'tab_'.$table.'_load_ctlf'} = $table.'_ctlf' if !defined
$params{'tab_'.$table.'_load_ctlf'};
        @ctlf = split(/,/, $params{'tab_'.$table.'_load_ctlf'});
        $err = "ctlf" if ((@ctlf > 1) && (@ctlf != $dp));
        @logf = split(/,/, $params{'tab_'.$table.'_load_logf'});
        $err = "logf" if ((@logf > 1) && (@logf != $dp));
        $lfexist = 1 if @logf > 0;
        @badf = split(/,/, $params{'tab_'.$table.'_load_badf'});
        $err = "badf" if ((@badf > 1) && (@badf != $dp));
        $bfexist = 1 if @badf > 0;
        @datf = split(/,/, $params{'tab_'.$table.'_load_datf'});
        $err = "datf" if ((@datf > 1) && (@datf != $dp));
        $dafexist = 1 if @datf > 0;
        @disf = split(/,/, $params{'tab_'.$table.'_load_disf'});
        $err = "disf" if ((@disf > 1) && (@disf != $dp));
        $difexist = 1 if @disf > 0;

        @filf = ();
        $params{'tab_'.$table.'_load_filf'} = $params{'scurts.'_datafiles'} if
($params{'tab_'.$table.'_load_filf'} == /alltsdatafiles/);
        if ($params{'tab_'.$table.'_load_filf'} == /alltsdatafiles/)
        {
            $params{'tab_'.$table.'_load_filf'} = $params{'scurts.'_datafiles'}
        }
        @filf = split(/,/, $params{'tab_'.$table.'_load_filf'}) if defined
$params{'tab_'.$table.'_load_filf'};
        # relax the requirement a little bit by allowing round robinning
        $numfil = @filf;
        #if ((($numfil > 1) && ($numfil < $dp) && (($dp % $numfil) == 0)) {
            if ((($numfil > 1) && ($numfil < $dp)) {
                $p = $dp / $numfil;
                $fil = "";
                for ($i = 0; $i < $dp; $i++)
                {
                    if ($i == 0) {
                        $fil = $params{'tab_'.$table.'_load_filf'};
                    } else {
                        $fil = join(',', $fil, $params{'tab_'.$table.'_load_filf'});
                    }
                }
                @filf = split(/,/, $fil);

                $err = "filf" if ((@filf > 1) && ($dp != @filf));
                $ffexist = 1 if @filf > 0;
                if ($err)
                {
                    print "Error with load parameters for table $table\n";
                    print " degree is $dp, $err list has bad number of elements\n";
                    return;
                }
                $susels = 1 if ((($dp > 1) && (@datf == 1) &&
$params{'tab_'.$table.'_load_datf'} != /\#/));

        # expand all of the xxxf arrays to arrays of size $dp
        @ff = ();
        for ($i=0; $i < $dp; $i++)
        {
            $iplus = $i + 1;
            if (@ctlf > 1)
            {
                @cf[$i] = @ctlf[$i];
            }
            else
            {
                @cf[$i] = @ctlf[0];
                @cf[$i] = s/\#/$iplus/g;
            }
            if (@logf > 1)
            {
                @lf[$i] = @logf[$i];
            }
            else
            {
                @lf[$i] = @logf[0];
            }
        }
    }
}

```

```

    {
        @lfl[$i] = @logf[0];
        @lfl[$i] -- s/\#/$iplus/g;
    }
    if (@badf > 1)
    {
        @bfl[$i] = @badf[$i];
    }
    else
    {
        @bfl[$i] = @badf[0];
        @bfl[$i] -- s/\#/$iplus/g;
    }
    if (@datf > 1)
    {
        @daf[$i] = @datf[$i];
    }
    else
    {
        @daf[$i] = @datf[0];
    }
    # if load_degpar is larger than the number partitions
    # and we are using pipes,
    # then load_degpar has to be a multiple of the number
    # of partition and load_degpar/#partitions sets of
    # pipes will be setup.
    # The goal is to maintain only one pipe/flatfile per
    # sqldr.

    if (($params{'tab_'. $table.'_#part'} > 0) &&
        ($dp > $params{'tab_'. $table.'_#part'})) {
        if ($dp % $params{'tab_'. $table.'_#part'} == 0) {
            if (@datf[0] == /\#.*\#/) {
                # set number
                $num=int($i/$params{'tab_'. $table.'_#part'}) + 1;
                # partition number
                $pnum=$i%$params{'tab_'. $table.'_#part'} + 1;
                @daf[$i] = @datf[0];
                @daf[$i] -- s/\#/$num/;
                @daf[$i] -- s/\#/$pnum/;
            } else {
                @daf[$i] = @datf[0];
                @daf[$i] -- s/\#/$iplus/g;
            }
        }
        else
        {
            print "Error: #partitions for table $table does not divide load_degpar for the
            table.\n";
            print "degree is $dp, number of partitions is $params{'tab_'. $table.'_#part'}.\n";
            return;
        }
    }
    else
    {
        # assign
        @daf[$i] = @datf[0];
        @daf[$i] -- s/\#/$iplus/g;
    }
}

if (@disf > 1)
{
    @dif[$i] = @disf[$i];
}
else
{
    @dif[$i] = @disf[0];
    @dif[$i] -- s/\#/$iplus/g;
}

if (@filf > 1)
{
    @ffl[$i] = @filf[$i];
}
else
{
    # round robinning - temp fix
    @ffl[$i] = @filf[0] if (@filf);
    if (defined $params{$params{'tab_'. $table.'_ts'}. '_#files'}) {
        $numfil = $params{$params{'tab_'. $table.'_ts'}. '_#files'};
        $filnum = $i % $numfil + 1;
        @ffl[$i] -- s/\#/$filnum/g;
    } else {
        @ffl[$i] -- s/\#/$iplus/g;
    }
}

$control = sprintf ("%s%s", $params{'load_controlfile_area'}, @cf[0]);
$control -- s/\#/$ENV{'ORACLE_HOME'}/g;

$ff_path = $params{'load_dbgen_'. $table.'_output_prefix'};
$numchild = $params{'load_dbgen_'. $table.'_option_C'};
$name = $params{'tab_'. $table.'_partnames'};
@name = split(/,/, $name);

# create the controlfiles (fixed-length fields or delimited records)
# if ($params{'skip_mk_idctlfs'} != /stable/)
# {
    $npart = (defined $params{'tab_'. $table.'_#part'}) ?
    $params{'tab_'. $table.'_#part'} : 1;
    for ($ictl=1; $ictl<=$npart; $ictl++) {
        open (CTFILE, ">$control");
        print CTFILE "---\n";
        print CTFILE "---- $table.ctl for delimited records\n" if ($ud ==
1);
        print CTFILE "---- $table.ctl for fixed-length fields\n" if ($ud ==
0);
        if (!($pre72) && defined $params{'tab_'. $table.'_loadextent'})
        {
            print CTFILE "options\n";
            print CTFILE "\n";
            print CTFILE "storage = (initial
$params{'tab_'. $table.'_loadextent'} next
$params{'tab_'. $table.'_loadextent'})\n";
            print CTFILE "\n";
        }
        else
        {
            print CTFILE "unrecoverable\n" if ($params{'load_unrecoverable'} ==
/[Tt][Rr][Uu][Ee]);
            print CTFILE "load\n";
            print CTFILE "-- This is where INFILE should go.\n";
            if (!($up) && ($dbgp)) {
                for ($jpart=1; $jpart<=$numchild; $jpart++) {
                    print CTFILE "INFILE '." . $ff_path . $jpart . ". $ictl." "\n";
                }
            }
            print CTFILE "$params{'load_insert_type'}\n";
            print CTFILE "into table $table\n";
            if ($dgp) {
                print CTFILE "partition ($pname{$ictl-1})\n";
            }
            print CTFILE "$params{'load_insert_type'}\n";
            print CTFILE "fields terminated by
$params{'load_field_terminator'}\n" if ($ud == 1);
            print CTFILE "\n";
            @tab_collist = split(/,/, $params{'tab_'. $table.'_columns'});
            while ($col = shift(@tab_collist))
            {
                (@tab_collist == 0) ? ($addcomma = "") : ($addcomma = ",");
                $pos = "";
                $pos = sprintf ("position
($s)", $params{'tab_'. $table.'_'. $col.'_pos'}) if ($ud == 0);
                $ctline = sprintf ("%s %s %s $addcomma", $col, $pos,
$params{'tab_'. $table.'_'. $col.'_loadcolx'});
                print CTFILE "$ctline\n";
            }
            print CTFILE "\n";
            close (CTFILE);
            if ($npart > 1) {
                $newi=$ictl+1;
                $control -- s/$table$ictl/$table$newi/;
            }
        }
    }

# create all of the "executable" load sections
# These variables will hold for all loaders for a given table
$upwd = "";
$parbool = "";
$dirbool = "";
$silent = "";
$dismax = "";
$file = "";
$errors = "";
$rows = "";
$bsize = "";
$load = "";
$skip = "";
$dbgopt = "";
$dbgtab = "";

if ($up) {
    if ($table == /lineitem/)
    {
        $dbgtab = "L";
    }
    elsif ($table == /orders/)
    {
        $dbgtab = "O";
    }
    elsif ($table == /partsupp/)
    {
        $dbgtab = "S";
    }
    elsif ($table == /parts/)
    {
        $dbgtab = "P";
    }
    elsif ($table == /customer/)
    {
        $dbgtab = "C";
    }
    elsif ($table == /supplier/)
    {
        $dbgtab = "s";
    }
    elsif ($table == /nation/)
    {
        $dbgtab = "n";
    }
    elsif ($table == /region/)
    {
        $dbgtab = "r";
    }
    $dbgopt = "-T ".$dbgtab." ";
    $dbgopt = "-s ".$params{'scale_factor'} . " ";
    $dbgopt = "-C ";
    $dbgopt = ((defined $params{'load_dbgen_'. $table.'_option_C'}) ?
$params{'load_dbgen_'. $table.'_option_C'} :
$params{'load_dbgen_def_option_C'});
    if ($dgp) {
        $dbgopt = "-p -i ";
        $dbgopt = ((defined $params{'load_dbgen_'. $table.'_input_params'}) ?
$params{'load_dbgen_'. $table.'_input_params'} :
$params{'load_dbgen_def_input_params'});
    }

    $upwd = "$params{'user'}/$params{'passwd'}";
    $parbool = "parallel=true" if (($params{'tab_'. $table.'_load_parallel'}
== /true/) || ($dp > 1));
    $dirbool = "direct=true" if (($params{'tab_'. $table.'_load_direct'} ==
/[Tt][Rr][Uu][Ee]) || $parbool);
    $silent = "silent=$params{'tab_'. $table.'_load_silent'}" if defined
$params{'tab_'. $table.'_load_silent'};
    $dismax = "discardmax=$params{'tab_'. $table.'_load_dismax'}" if defined
$params{'tab_'. $table.'_load_dismax'};
    $errors = "errors=$params{'tab_'. $table.'_load_errors'}" if defined
$params{'tab_'. $table.'_load_errors'};
    $rows = "rows=$params{'tab_'. $table.'_load_rows'}" if defined
$params{'tab_'. $table.'_load_rows'};
}

```

```

$bsize = "bindsize=$params{'tab_'. $table.'_load_bsize'}" if defined
$params{'tab_'. $table.'_load_bsize'};

if ($usels)
{
    $split = int($params{'tab_'. $table.'_#rows'} / $dp);
    $extra = int($params{'tab_'. $table.'_#rows'} % $dp);
    $loadval = $split;
    $skipval = 0;
}

# if using pipes, create the pipes

if ($up)
{
    for ($i=0; $i<$dp; $i++) {
        $cmd = sprintf("rm -f %s%snmknod %s%s
p\n", $params{'load_flatfile_area'}, @daf[$i], $params{'load_flatfile_area'}, @daf[$i]);
        # $cmd = sprintf("%s%s p\n", $params{'load_flatfile_area'}, @daf[$i]);
        &dump1("**sh");
        &dump0("**wait");

        # start the dbgens

        if ($dgp) {
            $nset = int($dp / $params{'tab_'. $table.'_#part'});
            $nperset = int($dbgp / $nset);

            for ($i=0; $i<$dbgp; $i++) {
                $cmd = $params{'load_flatfile_area'} . "\n";
                $cmd2 = sprintf("-f %s -S %d", $dbgopt, ($i + 1));
                if ($dgp) {
                    # if the output prefix name is specified with #
                    # divide output file names into $dp/#numpart sets
                    $fpre = (defined $params{'load_dbgen', $table.'_output_prefix'}) ?
                    $params{'load_dbgen', $table.'_output_prefix'} :
                    $params{'load_dbgen_def_output_prefix'};
                    if ($fpre =~ /\#/) {
                        $snum = int($i/$nperset) + 1;
                        $fpre =~ s/\#/$snum/g;
                    }
                    $cmd2 .= sprintf("-o %s ", $fpre);
                }
                $cmd2 .= "\n";
                &dump2("**dbgen");
            }
        }

        $control = "";
        $log = "";
        $bad = "";
        $data = "";
        $discard = "";
        $file = "";
        $load = "";
        $loadval = "";
        $skip = "";

        for ($i=0; $i < $dp; $i++)
        {
            &advmulti();

            $control = sprintf (" control=%s", $params{'load_controlfile_area'},
            @cf[$i]);
            $control = sprintf (" control=host%s", $params{'load_controlfile_area'}, @cf[$i]) if ($params{'special_machine'} eq
            'hcube');
            $log = sprintf (" log=%s", $params{'load_otherfile_area'}, @lf[$i]);
            if ($lfexist); $bad = sprintf (" bad=%s", $params{'load_otherfile_area'}, @bf[$i]);
            if ($bfexist); $data = sprintf (" data=%s", $params{'load_flatfile_area'},
            @daf[$i]);
            if ($dafexist); $discard = sprintf (" discard=%s", $params{'load_otherfile_area'},
            @dif[$i]);
            if ($difexist); $file = sprintf (" file=%s", $params{'load_otherfile_area'}, @ff[$i]);
            if ($ffexist);

            if ($usels)
            {
                $loadval = $loadval + 1 if ($extra > $i);
                $load = sprintf (" load=%d", $loadval);
                $skip = sprintf (" skip=%d", $skipval);
                $skipval = $skipval + $loadval;
                $loadval = $split;
            }

            if ($up)
            {
                $load = " load=999999999";
            }

            $cmd = sprintf
            ("%s%s%s%s%s%s%s%s%s%s%s\n", $upwd, $control, $log, $bad, $data, $discard, $dis
            max, $skip, $load, $errors, $rows, $bsize, $silent, $dirbool, $parbool, $file);
            &dump1("**load");
            &dump0("**wait") if (!(($parbool) && (!defined
            $params{'load_no_waits'})));
            &dump0("**wait") if (!defined $params{'load_no_waits'});
        }

        if ($up)
        {
            for ($i=0; $i<$dp; $i++) {
                $cmd = sprintf("rm -f %s%s\n", $params{'load_flatfile_area'}, @daf[$i]);
                &dump1("**sh");
            }
            &dump0("**wait");
        }
    }
}

sub plcre
{
    {
        &dump0("#####");
        &dump0("# NT Raw Partition and Setlink Creation Phase");
        &time0("Begin NT Raw Partition and Setlink creation");

        # create NT Raw Partition and Setlink input files
        $lnksfile = $params{'plcre_setlinks_input_file'};
        open (LNKSPFILE, ">$lnksfile");
        $drivenum = $params{'plcre_drivenums'};
        $md = $drivenum =~ tr/,/,/;
        @drivenum = split(/,/, $params{'plcre_drivenums'});
        if ($params{'plcre_recreate_extended_partitions'} =~ /[tT][rR][uU][eE]/))
        {
            foreach $drivenum (@drivenum)
            {
                &recreate_drive_extended_part;
            }
            if (defined $params{'plcre_log_drivenum'})
            {
                &recreate_drive_extended_part;
            }
        }
        $io_control_files = $params{'io_control_files'};
        $io_control_files =~ s/[\\(\\*\\|\\|\\|)]*/g;
        @io_control_files = split(/,/, $io_control_files);
        $d = 0;
        foreach $file (@io_control_files)
        {
            $size = $params{'plcre_control_file_size'};
            $size =~ s/[Mm]*/g;
            &create_drive_part_file;
        }
        foreach $ts_entry (@ts_all)
        {
            &create_drive_part("$ts_entry") if ($ts_entry =~ /ts_sys/ || $ts_entry =~
            /ts_log/);
        }
        $d = 0;
        foreach $ts_entry (@ts_most)
        {
            &create_drive_part("$ts_entry");
        }
        close (LNKSPFILE);
        &dump0("**wait");
        &time0("Done NT Raw Partition creation");
        $cmd = "setlinks /F:" . $lnksfile . "\n";
        &dump1("**sh");
        &dump0("**wait");
        &time0("Done Setlink links creation");
        &time0("Done NT Raw Partition and Setlink creation");
    } # end of plcre

    sub scrcr
    {
        &dump0("#####");
        &dump0("# Schema Creation Phase");
        &time0("Begin schema creation");

        # Create user
        if ($params{'skip_create_user'} != /true/)
        {
            &dump0("# creating $params{'user'} user");
            $cmd .= "drop user $params{'user'} cascade;\n";
            $cmd .= "grant $params{'privileges'}\n";
            $cmd .= " to $params{'user'} identified by $params{'passwd'};\n";
            $cmd .= "@?/rdbsms/admin/utlxlplan;\n" if (($os cmp "unix") == 0);
            &dump1("**sql");
            &dump0("**wait");
        }

        # Create data tablespaces (including datafiles and tables)
        &dump0("# creating data tablespaces, datafiles, and tables");

        # create tablespaces with initial datafile
        foreach $ts_entry (@ts_data)
        {
            &create_ts("$ts_entry");
        }
        foreach $ts_entry (@ts_index)
        {
            &create_ts("$ts_entry");
        }
        foreach $ts_entry (@ts_temp)
        {
            $ts_temporary = "temporary" if !$pre73;
            &create_ts("$ts_entry");
        }
        &dump0("**wait");

        # create remaining datafiles for the tablespaces
        foreach $ts_entry (@ts_data)
        {
            &add_dfs("$ts_entry");
        }
        foreach $ts_entry (@ts_index)
        {
            &add_dfs("$ts_entry");
        }
        if ($params{'skip_ts'} != /temp/)
        {
            foreach $ts_entry (@ts_temp)
            {
                &add_dfs("$ts_entry");
            }
        }
    }
}

```

```

&dump0("wait");

if ($params{'skip_create_tables'} != /true/)
{
    &create_clusters();
    &dump0("wait");
    &create_objects('tab',@tab_list) if defined(@tab_list);
    &dump0("wait");
}

if ($params{'skip_ts'} != /temp/)
{
    # Alter user's temporary tablespace
    &dump0("# altering $params{'user'}'s temporary tablespace");
    $ts_entry = shift (@ts_temp);
    $cmd = "alter user $params{'user'} temporary tablespace $ts_entry;\n";
    &dump1("sql");
    &dump0("wait");
}

if ($params{'skip_ts'} != /data/)
{
    # Alter user's default tablespace
    &dump0("# altering $params{'user'}'s default tablespace");
    $ts_entry = shift (@ts_data);
    $cmd = "alter user $params{'user'} default tablespace $ts_entry;\n";
    &dump1("sql");
    &dump0("wait");
}

&time0("Done schema creation");
}

sub sctso
{
    if ($phasetest != /sccre/)
    {
        &dump0("#####");
        &dump0("# Schema Creation Phase - datafiles only (no tables or users)");

        # Create data tablespaces (including datafiles)
        &dump0("# creating data tablespaces, datafiles");

        # create tablespaces with initial datafile
        foreach $ts_entry (@ts_data)
        {
            &create_ts("$ts_entry");
        }
        foreach $ts_entry (@ts_index)
        {
            &create_ts("$ts_entry");
        }
        foreach $ts_entry (@ts_temp)
        {
            $ts_temporary = "temporary" if !$pre73;
            &create_ts("$ts_entry");
        }
        &dump0("wait");

        # create remaining datafiles for the tablespaces
        foreach $ts_entry (@ts_data)
        {
            &add_dfs("$ts_entry");
        }
        foreach $ts_entry (@ts_index)
        {
            &add_dfs("$ts_entry");
        }
        if ($params{'skip_ts'} != /temp/)
        {
            foreach $ts_entry (@ts_temp)
            {
                &add_dfs("$ts_entry");
            }
        }
        &dump0("wait");

        if ($params{'skip_ts'} != /temp/)
        {
            # Alter user's temporary tablespace
            # &dump0("# altering $params{'user'}'s temporary tablespace");
            # $ts_entry = shift (@ts_temp);
            # $cmd = "alter user $params{'user'} temporary tablespace $ts_entry;\n";
            # &dump1("sql");
            # &dump0("wait");
        }
    }
}

sub scuto
{
    if ($phasetest != /sccre/)
    {
        &dump0("#####");
        &dump0("# Schema Creation Phase - User and Tables ONLY (no datafiles)");
        &time0("Begin creating user and tables (no datafiles)");

        if ($params{'skip_create_user'} != /true/)
        {
            &dump0("# creating $params{'user'} user");
            $cmd = "drop user $params{'user'} cascade;\n";
            $cmd = "grant $params{'privileges'}\n";
            $cmd = "to $params{'user'} identified by $params{'passwd'};\n";
            &dump1("sql");
        }
        &dump0("wait");
    }
}

&create_clusters();
&dump0("wait");
&create_objects('tab',@tab_list) if defined(@tab_list);
&dump0("wait");

if ($params{'skip_ts'} != /temp/)
{
    # Alter user's temporary tablespace
    &dump0("# altering $params{'user'}'s temp tablespace");
    $ts_entry = shift (@ts_temp);
    $cmd = "alter user $params{'user'} temporary tablespace $ts_entry;\n";
    &dump1("sql");
    &dump0("wait");
}

# Alter user's default tablespace
if ($params{'skip_ts'} != /data/)
{
    &dump0("# altering $params{'user'}'s default tablespace");
    $ts_entry = shift (@ts_data);
    $cmd = "alter user $params{'user'} default tablespace $ts_entry;\n";
    &dump1("sql");
    &dump0("wait");
}

&time0("Done creating user and tables (no datafiles)");
}

sub scuvo
{
    if ($phasetest != /scuvo/)
    {
        &dump0("#####");
        &dump0("# Schema Creation Phase - Views ONLY (no datafiles)");

        if (defined(@tablelog_list))
        {
            &dump0("#####");
            &dump0("# First I will create the materialized log for the base tables");
            &create_objects('viewlog',@tablelog_list);
            &dump0("wait");
        }

        &dump0("#####");
        &dump0("# Now I can create the views with the corresponding materialized logs");

        $cmd = "connect $params{'user'}/$params{'passwd'};\n";

        foreach $view (@view_list)
        {
            @viewlogs = split(/,/, $params{'view_' . $view . '_viewlogs'});
            foreach $viewlog (@viewlogs)
            {
                $obj_type = 'viewlog';
                $object = $viewlog;
                if (!defined($params{'created_' . $object}))
                {
                    $params{'created_' . $object} = $object;
                    &create_object;
                }
                $obj_type = 'view';
                $object = $view;
                &create_object;
            }
        }
        &dump1("sql");

        # &create_objects('view',@view_list) if defined(@view_list);
        #&create_objects('viewlog',@viewlog_list) if defined(@tablelog_list);

        &dump0("wait");
    }
}

sub dapop ## copied from Frank's version
{
    &dump0("#####");
    &dump0("# Database Population Phase");
    &time0("Begin data population");

    $cur_inst = 1 if $multi; # for possible locality on SP2
    ($params{'load_type'} =~ /delim/) ? ($sud = 1) : ($sud = 0);
    $params{'load_tables'} = $params{'tab_tables'} if !defined($params{'load_tables'});
    @load_tablelist = split(/,/, $params{'load_tables'});
    foreach $table (@load_tablelist)
    {
        &time0("Begin $table load");

        $curts = $params{'tab_' . $table . '_ts'};
        $sdp = $params{'tab_' . $table . '_load_degpar'};
        $susels = 0;
        $lfexist = 0;
        $bfexist = 0;
        $dafexist = 0;
        $difexist = 0;
        $ffexist = 0;

        $sup = 0;
        $dgp = 0;
        $dbgpar = 1;

        # see if we are using pipes
    }
}

```

```

Sup = 1 if ($params{'load_use_pipes'}) == /$table/);
$sdgp = 1 if ($params{'load_dbgen_partition'}) == /$table/;
$dbdgp = $params{'load_dbgen_def_option_C'} if defined
$params{'load_dbgen_def_option_C'};
$dbdgp = $params{'load_dbgen.'.$table.'.option_C'} if defined
$params{'load_dbgen.'.$table.'.option_C'};

$params{'tab'.'.$table.'.load_ctlf'} = $table.'.ctlf' if !defined
$params{'tab'.'.$table.'.load_ctlf'};
@ctlf = split(/./, $params{'tab'.'.$table.'.load_ctlf'});
$err = "ctlf" if ((@ctlf > 1) && (@ctlf != $dp));
$logf = split(/./, $params{'tab'.'.$table.'.load_logf'});
$err = "log" if ((@logf > 1) && (@logf != $dp));
$lfexist = 1 if @logf > 0;
@badf = split(/./, $params{'tab'.'.$table.'.load_badf'});
$err = "bad" if ((@badf > 1) && (@badf != $dp));
$bfexist = 1 if @badf > 0;
@daf = split(/./, $params{'tab'.'.$table.'.load_daf'});
$err = "dat" if ((@daf > 1) && (@daf != $dp));
$dafexist = 1 if @daf > 0;
@disf = split(/./, $params{'tab'.'.$table.'.load_disf'});
$err = "dis" if ((@disf > 1) && (@disf != $dp));
$difexist = 1 if @diff > 0;
@filf = ();

## NT Port
## exapnd all files for all partitions
if ($params{'tab'.'.$table.'.load_filf'}) == ~/alltsdatafiles/ )
{
    if ($params{'tab'.'.$table.'.part_ts'}) == ~/^(.*)%$/ && $dgp == 1)
    {
        $curts = "";
        $params{'tab'.'.$table.'.part_ts'} == ~/^(.*)%$/;
        $tsnam = $1;
        $params{'tab'.'.$table.'.load_filf'} = "";
        $nums = $params{'tab'.'.$table.'.#part'};
        $numfil = $params{$params{'tab'.'.$table.'.ts'}}.'_#files';

for ($j = 1; $j <= $nums; $j++)
{
    $tspnam = sprintf("%s%d", $tsnam, $j);
    if (defined $params{$tspnam.'_areas'})
    {
        $tsareas = $params{$tspnam.'_area'} . ", " . $params{$tspnam.'_areas'};
    }
    else
    {
        $tsareas = $params{$tspnam.'_area'};
    }
    for ($i = 1; $i < $numfil; $i++)
    {
        $tsareas = $tsareas . ", " . $params{$tspnam.'_area'};
    }
    $tsarea[$j] = [ split(/./, $tsareas) ];
}

for ($i = 1; $i <= $numfil; $i++)
{
    for ($j = 1; $j <= $nums; $j++)
    {
        $tspnam = sprintf("%s%d", $tsnam, $j);
        (($i == $numfil) && ($j == $nums)) ?
            ($addcomma = "") : ($addcomma = "," );
        if ($tsarea[$j][$i-1] =~ /\|\|\|.\|\|/)
        {
            $nextfile = sprintf("%s%s_d.dbf%s", $tsarea[$j][$i-1], $tspnam, $i, $addcomma);
        }
        else
        {
            # Temporay walkaround for 804 & SQLLDR bugs
            # $nextfile = sprintf("%s%s_d%s", $tsarea[$j][$i-1], $tspnam, $i, $addcomma);
            if (defined $params{'tab'.'.$table.'.part_lfn'})
            {
                $nextfile =
sprintf("\|\|\|\|\|.\\||\|s%d_t%s", $params{'tab'.'.$table.'.part_lfn'}, $j, $i, $addcomm
a);
            }
            else
            {
                $nextfile = sprintf("\|\|\|\|\|.\\||\|s_d%s", $tspnam, $i, $addcomma);
            }
            $params{'tab'.'.$table.'.load_filf'} .= $nextfile;
        }
    }
}
else
{
    $params{'tab'.'.$table.'.load_filf'} = $params{$curts.'_datafiles'};
}
}
elseif ($params{'tab'.'.$table.'.load_filf'}) == ~/^(.*)%$/ )
{
    $curts = "";
    $params{'tab'.'.$table.'.load_filf'} == ~/^(.*)%$/;
    $tsnam = $1;
    $params{'tab'.'.$table.'.load_filf'} = "";
    $tsareas = $params{$tsnam.'_area'} . ", " . $params{$tsnam.'_areas'};
    @tsarea = split(/./, $tsareas);
    $numfil = @tsarea;
    @dfile = split(/./, $params{$tsnam.'_datafiles'});
    $numfil = @dfile if ($numfil < @dfile);
for ($i = 0; $i < $numfil; $i++)
{
    ($i == $numfil - 1) ? ($addcomma = "") : ($addcomma = "," );
    if (@tsarea[0] =~ /\|\|\|.\|\|/)
    {
        $nextfile = sprintf("%s%s.dbf%s", @tsarea[$i], $dfile[$i], $addcomma);
    }
    else
    {
        $nextfile = sprintf("%s%s.dbf%s", @tsarea[$i], $dfile[$i], $addcomma);
    }
}

# Temporay workaroud of 804 & SQLLDR bugs
# $nextfile = sprintf("%s%s", @tsarea[$i], $dfile[$i], $addcomma);
$nextfile = sprintf("\|\|\|\|\|.\\||\|s%s", $dfile[$i], $addcomma);
}
}

## NT End

@filf = split(/./, $params{'tab'.'.$table.'.load_filf'}) if defined
$params{'tab'.'.$table.'.load_filf'};
# relax the requirement a little bit by allowing round robining
$numfil = @filf;
## NT Port (fix the case for partition tables which has only one file in each
partition
##
##--
##      if (($numfil > 1) && ($numfil < $dp) && (($dp % $numfil) == 0)) {
##          $p = $dp / $numfil;
##--
if (((($numfil > 1) && ($numfil < $dp) && (($dp % $numfil) == 0)) ||
(($dgp == 1) && ($dp == 1)))
{
    $p = 1;
    $p = ($dp / $numfil) if ($dp > 1);
    ## NT End
    $fil = "";
    for ($i = 0; $i < $p; $i++)
    {
        if ($i == 0) {
            $fil = $params{'tab'.'.$table.'.load_filf'};
        } else {
            $fil = join(',', $fil, $params{'tab'.'.$table.'.load_filf'});
        }
    }
    @filf = split(/./, $fil);
}

$err = "fil" if ((@filf > 1) && ($dp != @filf) && ($dp != 1));
$ffexist = 1 if @filf > 0;
if ($err)
{
    print "Error with load parameters for table $table\n";
    print " degree is $dp, $err list has bad number of elements\n";
    return;
}
$susels = 1 if (((($dp > 1) && (@daf == 1) &&
$params{'tab'.'.$table.'.load_daf'}) != /\#//));

# expand all of the xxxf arrays to arrays of size $dp

@ff = ();

$dp = @filf if ((@dp == 1) && ($dgp == 1));

for ($i=0; $i < $dp; $i++)
{
    $iplus = $i + 1;
    if (@ctlf > 1)
    {
        @cf[$i] = @ctlf[$i];
    }
    else
    {
        @cf[$i] = @ctlf[0];
        @cf[$i] -- s/\#/$iplus/g;
    }
    if (@logf > 1)
    {
        @lf[$i] = @logf[$i];
    }
    else
    {
        @lf[$i] = @logf[0];
        @lf[$i] -- s/\#/$iplus/g;
    }
    if (@badf > 1)
    {
        @bf[$i] = @badf[$i];
    }
    else
    {
        @bf[$i] = @badf[0];
        @bf[$i] -- s/\#/$iplus/g;
    }
    if (@daf > 1)
    {
        @daf[$i] = @daf[$i];
    }
    else
    {
        # if load degpar is larger than the number partitions
        # and we are using pipes,
        # then load degpar has to be a multiple of the number
        # of partition and load_degpar/#partitions sets of
        # pipes will be setup.
        # The goal is to maintain only one pipe/flatfile per
        # sqldr.
        if (($params{'tab'.'.$table.'.#part'}) > 0) &&
($dp > $params{'tab'.'.$table.'.#part'}) {
            if ($dp % $params{'tab'.'.$table.'.#part'}) == 0 {
                # set number
                $nnum=int($i/$params{'tab'.'.$table.'.#part'}) + 1;
                # partiton number
                $pnnum=$i-$params{'tab'.'.$table.'.#part'} + 1;
                @daf[$i] = @daf[0];
                @daf[$i] -- s/\#/$nnum/g;
                @daf[$i] -- s/\#/$pnnum/g;
            }

```

```

} else {
    @daf[$i] = @datf[0];
    @daf[$i] =~ s/\#/$iplus/g;
}
}
else
{
    print "Error: #partitons for table $table does not divide load_degpar for the
    table.\n";
    print "degree is $dpp, number of partitions is $params{'tab_'. $table.'_#part'}. \n";
    return;
}
}
else
{
    # assign
    @daf[$i] = @datf[0];
    @daf[$i] =~ s/\#/$iplus/g;
}
}
if (@disf > 1)
{
    @dif[$i] = @disf[$i];
}
else
{
    @dif[$i] = @disf[0];
    @dif[$i] =~ s/\#/$iplus/g;
}
if (@filf > 1)
{
    @ff[$i] = @filf[$i];
}
else
{
    # round robining - temp fix
    @ff[$i] = @filf[0] if (@filf);
    if (defined $params{$params{'tab_'. $table.'_ts'}. '_#files'}) {
        $numfil = $params{$params{'tab_'. $table.'_ts'}. '_#files'};
        HMP print "Table \n";
        HMP print "Number of files \n";
        $filnum = $i % $numfil + 1;
        @ff[$i] = s/\#/$filnum/g;
    } else {
        @ff[$i] = s/\#/$iplus/g;
    }
}
$control = sprintf ("%s", $params{'load_controlfile_area'}, @cf[0]);
$control =~ s/\#/$ENV{'ORACLE_HOME'}/g;

# add code to create head and tail of control files

$controlh = sprintf ("%s", $params{'load_controlfile_area'}, $table);
$controlh =~ s/\#/$ENV{'ORACLE_HOME'}/g;

$controlt = sprintf ("%s", $params{'load_controlfile_area'}, $table);
$controlt =~ s/\#/$ENV{'ORACLE_HOME'}/g;

open (CTFILEH, ">$controlh");
open (CTFILET, ">$controlt");
print CTFILEH "---\n";
if (!($pre72) && defined $params{'tab_'. $table.'_loadextent'})
{
    print CTFILEH "options\n";
    print CTFILEH "\n";
    print CTFILEH "storage = (initial
$params{'tab_'. $table.'_loadextent'} next
$params{'tab_'. $table.'_loadextent'})\n";
    print CTFILEH "\n";
}
# print CTFILE "unrecoverable\n" if ($params{'load_unrecoverable'} ==
/[tT][rR][uU][eE]);
print CTFILEH "unrecoverable\n" if ($params{'load_unrecoverable'} ==
/[tT][rR][uU][eE]);

print CTFILEH "load\n";
print CTFILEH "-- This is where INFILE should go.\n";
# print CTFILEH "load_insert_type'\n";
print CTFILEH "into table $table\n";

print CTFILET "-- This is where PARTITION is specified.\n";
print CTFILET "load_insert_type'\n";
print CTFILET "fields terminated by $params{'load_field_terminator'}\n" if ($sud
== 1);
print CTFILET "\n";

@tab_collist = split(/,/, $params{'tab_'. $table.'_columns'});
while ($col = shift(@tab_collist))
{
    (@tab_collist == 0) ? ($addcomma = "") : ($addcomma = ",");
    $pos = "";
    $pos = sprintf ("position (%s)", $params{'tab_'. $table.'_'. $col.'_pos'}) if
($sud == 0);
    $colline = sprintf (" %s %s %s", $col, $pos,
$params{'tab_'. $table.'_'. $col.'_loadcolx'});
    print CTFILET "$colline\n";
}
print CTFILET "\n";
close (CTFILEH);
close (CTFILET);

# create the controlfiles (fixed-length fields or delimited records)
if ($params{'skip_mk_idctifs'} != /table/)
{
    open (CTFILE, ">$control");
    print CTFILE "----\n";
    print CTFILE "---- $table.ctl for delimited records\n" if ($sud == 1);
    print CTFILE "---- $table.ctl for fixed-length fields\n" if ($sud ==
0);
    print CTFILE "----\n";
}
if (!($pre72) && defined $params{'tab_'. $table.'_loadextent'})
{
    print CTFILE "options\n";
    print CTFILE "\n";
    print CTFILE "storage = (initial
$params{'tab_'. $table.'_loadextent'} next
$params{'tab_'. $table.'_loadextent'})\n";
    print CTFILE "\n";
}
# print CTFILE "unrecoverable\n" if ($params{'load_unrecoverable'} ==
/[tT][rR][uU][eE]);
print CTFILE "unrecoverable\n" if ($params{'load_unrecoverable'} ==
/[tT][rR][uU][eE]);

print CTFILE "load\n";
print CTFILE "-- This is where INFILE should go.\n";
# print CTFILE "load_insert_type'\n";
print CTFILE "into table $table\n";

print CTFILET "-- This is where PARTITION is specified.\n";
print CTFILET "load_insert_type'\n";
print CTFILET "fields terminated by $params{'load_field_terminator'}\n" if ($sud
== 1);
print CTFILET "\n";

@tab_collist = split(/,/, $params{'tab_'. $table.'_columns'});
while ($col = shift(@tab_collist))
{
    (@tab_collist == 0) ? ($addcomma = "") : ($addcomma = ",");
    $pos = "";
    $pos = sprintf ("position (%s)", $params{'tab_'. $table.'_'. $col.'_pos'}) if
($sud == 0);
    $colline = sprintf (" %s %s %s", $col, $pos,
$params{'tab_'. $table.'_'. $col.'_loadcolx'});
    print CTFILET "$colline\n";
}
print CTFILET "\n";
close (CTFILEH);
close (CTFILET);

# create the controlfiles (fixed-length fields or delimited records)
if ($params{'skip_mk_idctifs'} != /table/)
{
    open (CTFILE, ">$control");
    print CTFILE "----\n";
    print CTFILE "---- $table.ctl for delimited records\n" if ($sud == 1);
    print CTFILE "---- $table.ctl for fixed-length fields\n" if ($sud ==
0);
    print CTFILE "----\n";
}
if (!($pre72) && defined $params{'tab_'. $table.'_loadextent'})
{
    print CTFILE "options\n";
    print CTFILE "\n";
    print CTFILE "storage = (initial
$params{'tab_'. $table.'_loadextent'} next
$params{'tab_'. $table.'_loadextent'})\n";
    print CTFILE "\n";
}
# print CTFILE "unrecoverable\n" if ($params{'load_unrecoverable'} ==
/[tT][rR][uU][eE]);
print CTFILE "unrecoverable\n" if ($params{'load_unrecoverable'} ==
/[tT][rR][uU][eE]);

print CTFILE "load\n";
print CTFILE "-- This is where INFILE should go.\n";
# print CTFILE "load_insert_type'\n";
print CTFILE "into table $table\n";

print CTFILET "-- This is where PARTITION is specified.\n";
print CTFILET "load_insert_type'\n";
print CTFILET "fields terminated by $params{'load_field_terminator'}\n" if ($sud
== 1);
print CTFILET "\n";

@tab_collist = split(/,/, $params{'tab_'. $table.'_columns'});
while ($col = shift(@tab_collist))
{
    (@tab_collist == 0) ? ($addcomma = "") : ($addcomma = ",");
    $pos = "";
    $pos = sprintf ("position (%s)", $params{'tab_'. $table.'_'. $col.'_pos'}) if
($sud == 0);
    $colline = sprintf (" %s %s %s", $col, $pos,
$params{'tab_'. $table.'_'. $col.'_loadcolx'});
    print CTFILET "$colline\n";
}
print CTFILET "\n";
close (CTFILEH);
close (CTFILET);

# create all of the "executable" load sections
# These variables will hold for all loaders for a given table
$supwd = "";
$parbool = "";
$dbrbool = "";
$silent = "";
$dismax = "";
$file = "";
$errors = "";
$rows = "";
$bsize = "";
$load = "";
$skip = "";
$dbgopt = "";
$dbgtab = "";

if ($sup) {
    if ($table == /lineitem/)
    {
        $dbgtab = "L";
    }
    elsif ($table == /orders/)
    {
        $dbgtab = "O";
    }
    elsif ($table == /partsupp/)
    {
        $dbgtab = "S";
    }
    elsif ($table == /parts/)
    {
        $dbgtab = "P";
    }
    elsif ($table == /customer/)
    {
        $dbgtab = "C";
    }
    elsif ($table == /supplier/)
    {
        $dbgtab = "S";
    }
    elsif ($table == /nation/)
    {
        $dbgtab = "N";
    }
    elsif ($table == /region/)
    {
        $dbgtab = "R";
    }
    $dbgopt = "-T " . $dbgtab . " ";
    $dbgopt = "-s " . $params{'scale_factor'} . " ";
    $dbgopt = "-C ";
    $dbgopt = "((defined $params{'load_dbgen_'. $table.'_option_C'}) ?
$params{'load_dbgen_'. $table.'_option_C'} :
$params{'load_dbgen_def_option_C'});
    if ($dgp) {
        $dbgopt = "-p -i ";
        $dbgopt = "((defined $params{'load_dbgen_'. $table.'_input_params'}) ?
$params{'load_dbgen_'. $table.'_input_params'} :
$params{'load_dbgen_def_input_params'});
    }
}

$supwd = $params{'user'} / $params{'passwd'};
$parbool = "parallel=true" if ((($params{'tab_'. $table.'_load_parallel'}
== /true/) || ($dpp > 1));
$dbrbool = "direct=true" if ((($params{'tab_'. $table.'_load_direct'} ==
/[Tt][Rr][Uu][Ee]) || ($parbool));
$silent = "silent=$params{'tab_'. $table.'_load_silent'}" if defined
$params{'tab_'. $table.'_load_silent'};
$dismax = "discardmax=$params{'tab_'. $table.'_load_dismax'}" if defined
$params{'tab_'. $table.'_load_dismax'};
$errors = "errors=$params{'tab_'. $table.'_load_errors'}" if defined
$params{'tab_'. $table.'_load_errors'};
$rows = "rows=$params{'tab_'. $table.'_load_rows'}" if defined
$params{'tab_'. $table.'_load_rows'};
$bsize = "bindsize=$params{'tab_'. $table.'_load_bsize'}" if defined
$params{'tab_'. $table.'_load_bsize'};

if ($usels)
{
    $split = int($params{'tab_'. $table.'_#rows'} / $dpp);
}

```

```

        $extra = int($params{'tab_'. $table.'_'$_rows'}) % $dp;
        $loadval = $split;
        $skipval = 0;
    }

## NT Port (use dbgen to create flat files)
##
### if using pipes, create the pipes
##
##if($sup)
##{
##    for ($i=0;$i<$dp;$i++) {
##        $cmd = sprintf("%s%s p\n", $params{'load_flatfile_area'}, @daf[$i]);
##        &dump1("mknod");
##    }
##    &dump0("wait");
##    # start the dbgens
##
##    if ($dgp) {
##        $nset = int($dp / $params{'tab_'. $table.'_'$_part'});
##        $nperset = int($dbgp / $nset);
##    }
##
##    for ($i=0;$i<$dbgp;$i++) {
##        $cmd = $params{'load_flatfile_area'} . "\n";
##        $cmd2 = sprintf("%s -S %d", $dbopt, ($i + 1));
##        if ($dgp) {
##            # if the output prefix name is specified with #
##            # divide output file names into $dp/#numpart sets
##            $fpre = (defined $params{'load_dbgen', $table.'_'$_output_prefix'}) ?
##                $params{'load_dbgen', $table.'_'$_output_prefix'} :
##                $params{'load_dbgen_def_output_prefix'};
##            if ($fpre =~ /\#/) {
##                $snum = int($i/$nperset) + 1;
##                $fpre =~ s/\#/$snum/g;
##            }
##            $cmd2 .= sprintf("-o %s ", $fpre);
##        }
##
##        ### Added 4/28? - Ari
##        ## need to force in case pipe still there, and need to kick
##        ## off in the background from here
##        $cmd2 = "-f " . $cmd2;
##        $cmd2 .= " & ";
##
##        $cmd2 .= "\n";
##        &dump2("dbgen");
##    }
##
##    &dump0("wait");
## NT End

    for ($i=0; $i < $dp; $i++)
    {
        &vmulti();

        $control = sprintf (" control=%s", $params{'load_controlfile_area'},
@cf[$i]);
        $control = sprintf (" control=/host%s", $params{'special_machine'} eq
$params{'load_controlfile_area'}, @cf[$i]) if ($params{'special_machine'} eq
'hcube');
        $log = sprintf (" log=%s", $params{'load_otherfile_area'}, @lf[$i])
if $lfexist;
        $bad = sprintf (" bad=%s", $params{'load_otherfile_area'}, @bf[$i])
if $bfexist;
        $data = sprintf (" data=%s", $params{'load_flatfile_area'},
@daf[$i]) if $dafexist;
        $discard = sprintf (" discard=%s", $params{'load_otherfile_area'},
@dif[$i]) if $difexist;
        if ($ffexist)
        {
            if ($curts == "")
            {
                $file = sprintf (" file=%s", @ff[$i]);
            }
            else
            {
                $file = sprintf (" file=%s", $params{$curts.'_area'}, @ff[$i]);
            }
        }
        ## NT Port
        ## Change to UPPER case string as a workaround for SQLLDR
        $file =~ tr/a-z/A-Z/;
        ## NT End

        if ($usels)
        {
            $loadval = $loadval + 1 if ($extra > $i);
            $load = sprintf (" load=%d", $loadval);
            $skip = sprintf (" skip=%d", $skipval);
            $skipval = $skipval + $loadval;
            $loadval = $split;
        }

        ## if ($sup)
        ## {
        ## $load = " load=999999999";
        ## }

        $cmd = sprintf
("%s%s%s%s%s%s%s%s%s%s\n", $supwd, $control, $log, $bad, $data, $discard, $dis
max, $skip, $load, $errors, $rows, $bsize, $silent, $dirbool, $parbool, $file);
        &dump1("load");
        &dump0("wait") if (!($parbool) && (!defined
$params{'load_no_waits'}));
        &dump0("wait") if (!defined $params{'load_no_waits'});

##--if($sup)
##{
##    for ($i=0;$i<$dp;$i++) {
##        $cmd = sprintf("rm -f %s\n", $params{'load_flatfile_area'}, @daf[$i]);
##        &dump1("sh");
##    }
##    &dump0("wait");
##}

##--}
        &time0("End $table load");
    }

    $cmd = "shutdown\n"; #^^sd
    &dump1("sql");
    &dump0("wait");
    &time0("Done data population");
}

sub ixcre
{
    &dump0("#####");
    &dump0("# Index Creation Phase");
    &time0("Begin index creation");
    #^^su
    if (($os cmp "nt") == 0)
    {
        if ($params{'skip_ixcre_shut'} != /true/)
        {
            &dump0("wait");
            $cmd = "shutdown\n";
            $cmd .= "startup pfile=" . $params{'dbs_area'} . "init1" . $params{'user'} .
".ora\n";
            &dump1("sql");
            &dump0("wait");
        }
    }

    foreach $index (@indexlist)
    {
        if (defined $params{'ind_'. $index.'_'$_table'})
        {
            $sob_type = 'table';
            $sob_type = 'tab';
        }
        else
        {
            $sob_type = 'view';
            $sob_type = 'view';
        }
    }

    &time0("Begin creating index $index");
    &advmulti();
    $cmd .= "connect $params{'user'}/$params{'passwd'};\n";
    $cmd .= "drop index $index;\n";
    $cmd .= "create ";
    if ($params{'ind_'. $index.'_'$_unique'}) == /{t}[r][u][e]/
    {
        $cmd .= "unique ";
    }
    elseif ($params{'ind_'. $index.'_'$_bitmap'}) == /{t}[r][u][e]/
    {
        $cmd .= "bitmap ";
    }
    $cmd .= "index $index\n";
    if (defined $params{'ind_'. $index.'_'$_sob_type'})
    {
        $cmd = "on $params{'ind_'. $index.'_'$_sob_type}
($params{'ind_'. $index.'_'$_sob_type'} 'cols')\n";
        $indtab = $params{'ind_'. $index.'_'$_sob_type'};
    }
    else
    {
        $cmd = "on cluster $params{'ind_'. $index.'_'$_cluster'}\n";
    }
    $cmd .= "pctfree $params{'ind_'. $index.'_'$_f}%\n" if defined
$params{'ind_'. $index.'_'$_f'};
    $cmd .= "initrans $params{'ind_'. $index.'_'$_it}\n" if defined
$params{'ind_'. $index.'_'$_it'};
    $cmd .= "maxtrans $params{'ind_'. $index.'_'$_mt}\n" if defined
$params{'ind_'. $index.'_'$_mt'};
    if ($params{'compatible'}) == /7\./
    {
        $cmd = "unrecoverable\n" if ($params{'ind_'. $index.'_'$_unrecoverable'}) ==
/{t}[r][u][e]/;
    }
    elseif ($params{'compatible'}) == /8\./
    {
        $cmd = "nologging\n" if ($params{'ind_'. $index.'_'$_nolg'}) ==
/{t}[r][u][e]/;
    }
    $cmd = "compute statistics\n" if ($params{'ind_'. $index.'_'$_compstats'}) ==
/{t}[r][u][e]/;
    $cmd = "tablespace $params{'ind_'. $index.'_'$_ts}\n" if ((defined
$params{'ind_'. $index.'_'$_ts'}) && ($params{'skip_ts'}) != /index/);
    $cmd = "storage $params{'ind_'. $index.'_'$_storage'}\n" if defined
$params{'ind_'. $index.'_'$_storage'};
    $cmd = "reverse\n" if ($params{'ind_'. $index.'_'$_reverse'}) == /{t}[r][u][e]/;
    $cmd = "nosort\n" if ($params{'ind_'. $index.'_'$_nosort'}) ==
/{t}[r][u][e]/;
    if ((defined $params{'ind_'. $index.'_'$_pardeg'}) || (defined $params{'ind_
'. $index.'_'$_parinst'}))
    {
        $cmd = "parallel";
        if ($params{'ind_'. $index.'_'$_pardeg'}) == /{d}[e][f][a][u][l]/
        {
            $cmd = "\n";
        }
        else
        {
            $cmd = " (degree $params{'ind_'. $index.'_'$_pardeg'} "
if defined $params{'ind_'. $index.'_'$_pardeg'};
            $cmd = "instances $params{'ind_'. $index.'_'$_parinst'}"
if defined $params{'ind_'. $index.'_'$_parinst'};
            $cmd = $cmd . "\n";
        }
    }

    # deal with partitioning

    if ($params{'ind_'. $index.'_'$_partition'}) == /{l}[o][c][a][l]/
    {
        $numpart = $params{$sob_type.'_'$_indtab.'_'$_part'};
        $cmd .= "local";
    }
}

```

```

        if (defined $params{'ind_'. $indtab.'_partnames'})
        {
&exp_ind_part_1;
        }
        else
        {
$cmd .= "\n";
        }
    }
    elseif ($params{'ind_'. $index.'_partition'}) =~ /[gG][lL][oO][bB][aA][lL]/ {
$numpart = $params{'ind_'. $index.'_part'};
$cmd .= "global partition by range (";
&exp_ind_part_g;
    }

    $cmd .= ";";
    &dump1("sql");
}

# Constraints which are enabled after the load
foreach $const (@constlist)
{
    &time0("Begin creating constraint $const");
    &advmulti();

    $cmd .= "connect $params{'user'}/$params{'passwd'};\n";
    if ($params{'con_'. $const.'_constraint'}) =~ /null/ {
        @collist = split(/,/, $params{'con_'. $const.'_columns'});
        $cmd .= "alter table $params{'con_'. $const.'_'. $sob_type} ";
        $cmd .= "modify (";
        while ($col = shift(@collist))
        {
            (@collist == 0) ? ($addcomma = "") : ($addcomma = ",";);
            $cmd .= "$col $params{'con_'. $const.'_constraint'}$addcomma";
        }
        $cmd .= ");";
    }
    else
    {
        $cmd .= "alter table $params{'con_'. $const.'_'. $sob_type} drop
constraint $const;\n";
        $cmd .= "alter table $params{'con_'. $const.'_'. $sob_type} ";
        $cmd .= "add constraint $const;\n";
        $cmd .= "
$params{'con_'. $const.'_columns'} key
if ($params{'con_'. $const.'_has_index'}) =~ /[Tt][Rr][Uu][Ee]/ {
$cmd .= "using index";
        }
        if ($params{'con_'. $const.'_disable'}) =~ /[Tt][Rr][Uu][Ee]/ {
$cmd .= "disable;\n";
        }
        else {
$cmd .= ";";
        }
        if ($params{'con_'. $const.'_constraint'}) =~ /foreign/ {
$cmd .= "alter table $params{'con_'. $const.'_'. $sob_type} ";
$cmd .= "enable novalidate primary key;\n";
        }
        if ($params{'con_'. $const.'_constraint'}) =~ /foreign/ {
$cmd .= "alter table $params{'con_'. $const.'_'. $sob_type} ";
$cmd .= "enable primary key";
        }
        if ($params{'con_'. $const.'_has_index'}) =~ /[Tt][Rr][Uu][Ee]/ {
$cmd .= "";
        }
        else {
$cmd .= "using index\n";
$cmd .= "pctfree $params{'con_'. $const.'_'. $sob_type} if defined
$params{'con_'. $const.'_'. $sob_type}";
$cmd .= "initrans $params{'con_'. $const.'_it'}\n if defined
$params{'con_'. $const.'_it'}";
$cmd .= "maxtrans $params{'con_'. $const.'_mt'}\n if defined
$params{'con_'. $const.'_mt'}";
$cmd .= "tablespace $params{'con_'. $const.'_ts'}\n if ((defined
$params{'con_'. $const.'_ts'}) && ($params{'skip_ts'}) != /index/);
$cmd .= "storage $params{'con_'. $const.'_storage'}\n if defined
$params{'con_'. $const.'_storage'}";
$cmd .= "nosort\n if ($params{'con_'. $const.'_nosort'}) =~ /true/;
$cmd .= ";";
        }
    }
    &dump1("sql");
    &dump0("wait") if (($os cmp "nt") == 0);
    &time0("Done creating constraint $const");
}

# Alter DOP of indexes (NT support)
if (($os cmp "nt") == 0)
{
    &dump0("wait");
    &time0("Alter DOP of indexes");
    $cmd .= "connect $params{'user'}/$params{'passwd'};\n";

    foreach $index (@indexlist)
    {
        &advmulti();

        if (defined $params{'ind_'. $index.'_pardeg_alter'})
        {
            $cmd .= "alter index $index parallel";
            $cmd .= " (degree $params{'ind_'. $index.'_pardeg_alter'})";
            $cmd .= " instances $params{'ind_'. $index.'_parinst'};\n";
        }
    }
    &dump1("sql");
    &dump0("wait");
    &time0("Done altering DOP of indexes");
    &time0("Done index creation");
} # end of ixore

sub exp_ind_part_1
{
    @ind_partnames = ();
    $params{'ind_'. $index.'_part'} =

    $params{'sob_type.'_' . $params{'ind_'. $index.'_' . $sob_type.'_' . $part'}};

    if (!defined $params{'ind_'. $index.'_partnames'})
    {
        # if no partnames are specified, use a default part name
        for ($i = 1; $i <= $params{'ind_'. $index.'_part'}; $i++)
        {
            ($i == $params{'ind_'. $index.'_part'}) ? ($addcomma = "") :
            ($addcomma = ",";);
            $nextfile = sprintf("%s%s%d%s", $index, "_p", $i, $addcomma);
            $params{'ind_'. $index.'_partnames'} =
            $params{'ind_'. $index.'_partnames'} . $nextfile;
        }
    }

    @ind_partnames = split(/,/, $params{'ind_'. $index.'_partnames'});
    if ($ind_partnames[0] =~ /\#/)
    {
        $filenm = shift(@ind_partnames);
        $savenam = $filenm;
        $params{'ind_'. $index.'_partnames'} = "";
        # indtab defined in ixore
        for ($i = 1; $i <= $numpart; $i++)
        {
            $filenm =~ s/\#/$i/g;
            ($i == $numpart) ? ($addcomma = "") : ($addcomma = ",";);
            $params{'ind_'. $index.'_partnames'} =
            $params{'ind_'. $index.'_partnames'} . $filenm . $addcomma;
            $filenm = $savenam;
        }
        @ind_partnames = split(/,/, $params{'ind_'. $index.'_partnames'});
        print "Expanded $savenam to:\n$params{'ind_'. $index.'_partnames'}\n\n" if
        $verbose;
    }
    else {
        if (@ind_partnames != $numpart)
        {
            print "number of partitions $numpart from the base table $indtab\ndoes not
            equal to the number of partition names defined in ind_'. $index.'_partnames.\n";
            exit(-1);
        }
    }

    &process_ind_part_ts;

    # complete the local partition index statement
    if ($numpart > 0)
    {
        $cmd .= "\n";
        for ($i=0; $i < $numpart; $i++)
        {
            $cmd .= "partition ";
            $cmd .= $ind_partnames[$i] . "\n" if @ind_partnames;
            &process_part_params('ind', '%f', 'pctfree', $index,
            (@ind_partnames) ? $ind_partnames[$i] : "");
            &process_part_params('ind', '%u', 'pctused', $index,
            (@ind_partnames) ? $ind_partnames[$i] : "");
            &process_part_params('ind', 'it', 'initrans', $index,
            (@ind_partnames) ? $ind_partnames[$i] : "");
            &process_part_params('ind', 'mt', 'maxtrans', $index,
            (@ind_partnames) ? $ind_partnames[$i] : "");
            if ((@ind_partnames) &&
            (defined $params{'ind_'. $index.'_' . $ind_partnames[$i].'_ts'}))
            {
                $cmd .= 'tablespace ' .
                $params{'ind_'. $index.'_' . $ind_partnames[$i].'_ts'} . "\n";
            }
            &process_part_storage('ind', $index, $ind_partnames[$i]);
            }
            elsif ((@ind_partnames) &&
            defined $params{'ind_'. $index.'_part_ts'})
            {
                $cmd .= 'tablespace ' . $ind_part_ts[$i] . "\n";
            }
            &process_part_storage('ind', $index, $ind_partnames[$i]);
            }
            if (@ind_partnames &&
            defined $params{'ind_'. $index.'_' . $ind_partnames[$i].'_nolg'})
            {
                $cmd .= "nologging\n";
            }
            if ($params{'ind_'. $index.'_' . $ind_partnames[$i].'_nolg'})
            == /[tT][rR][uU][eE]/;
            }
            elsif (defined $params{'ind_'. $table.'_part_def_nolg'}) {
                $cmd .= "nologging\n";
            }
            if ($params{'ind_'. $index.'_part_def_nolg'}) =~
            /[tT][rR][uU][eE]/;
            }
            $cmd .= ((($i+1) == $numpart) ? "\n" : ",\n";
            }
        }
        $cmd .= "\n";
    }
} # end of exp_ind_part_1

sub exp_ind_part_g
{
    @ind_partnames = ();
    @pcollist = split(/,/, $params{'ind_'. $index.'_partcol'});
    $cmd .= "$params{'ind_'. $index.'_partcol'}\n\n";

    if (!defined $params{'ind_'. $index.'_partnames'})
    {
        # if no partnames are specified, use a default part name
        for ($i = 1; $i <= $params{'ind_'. $index.'_part'}; $i++)
        {
            ($i == $params{'ind_'. $index.'_part'}) ? ($addcomma = "") :
            ($addcomma = ",";);
            $nextfile = sprintf("%s%s%d%s", $index, "_p", $i, $addcomma);
            $params{'ind_'. $index.'_partnames'} =
            $params{'ind_'. $index.'_partnames'} . $nextfile;
        }
    }

    @ind_partnames = split(/,/, $params{'ind_'. $index.'_partnames'});

    if ($ind_partnames[0] =~ /\#/)

```

```

{
    $filenm = $ind_partnames[0];
    $savename = $filenm;
    $params{'ind'.'.index.'_partnames'} = "";
    # indtab defined in ixcre
    for ($i = 1; $i <= $numpart; $i++)
    {
        $filenm =~ s/\#/$i/g;
        ($i == $numpart) ? ($addcomma = "") : ($addcomma = "," );
        $params{'ind'.'.index.'_partnames'} =
        $params{'ind'.'.index.'_partnames'} . $filenm . $addcomma;
        $filenm = $savename;
    }
    @ind_partnames = split(/,/, $params{'ind'.'.index.'_partnames'});
    print "Expanded $savename to:\n$params{'ind'.'.index.'_partnames'}\n\n" if
    $verbose;
    } else {
    if (@ind_partnames != $numpart)
    {
        print "number of partitions $numpart from the base table $indtab\ndoes not
        equal to the number of partition names defined in ind_$index_partnames.\n";
        exit(-1);
    }
    }

    # process boundaries and tablespaces

    &process_ind_part_brys;
    &process_ind_part_ts;

    # complete the local partition index statement

    for ($i=0; $i < $numpart; $i++)
    {
        $cmd .= "partition " . $ind_partnames[$i] . " values less than ";
        if ($i==0) {
            $cmd .= "($table.'_#part')-1) {"
            $cmd .= "(MAXVALUE)\n";
        }
        else {
            $cmd .= "( " . $ind_part_vals[$i] . " )\n";
        }

        &process_part_params('ind','%', 'pctfree', $index, $ind_partnames[$i]);
        &process_part_params('ind','%', 'pctused', $index, $ind_partnames[$i]);
        &process_part_params('ind','it', 'initrans', $index, $ind_partnames[$i]);
        &process_part_params('ind','mt', 'maxtrans', $index, $ind_partnames[$i]);

        if (defined $params{'ind'.'.index.'_'. $ind_partnames[$i] . '_ts'})
        {
            $cmd .= 'tablespace ' .
            $params{'ind'.'.index.'_'. $ind_partnames[$i] . '_ts'} . "\n";
            &process_part_storage('ind', $index, $ind_partnames[$i]);
        }
        elseif (defined $params{'ind'.'.index.'_part_ts'})
        {
            $cmd .= 'tablespace ' . $ind_part_ts[$i] . "\n";
            &process_part_storage('ind', $index, $ind_partnames[$i]);
        }
    }
    if (defined $params{'ind'.'.index.'_'. $ind_partnames[$i] . '_nolg'})
    {
        $cmd .= "nologging\n";
        if ($params{'ind'.'.index.'_'. $ind_partnames[$i] . '_nolg'})
        == /[tT][rR][uU][eE]/;
    }
    elseif (defined $params{'ind'.'.table.'_part_def_nolg'})
    {
        $cmd .= "nologging\n";
        if ($params{'ind'.'.index.'_part_def_nolg'}) ==
        /[tT][rR][uU][eE]/;
    }
    }
    $cmd .= (($i+1) == $numpart) ? "" : ",\n";
    }
    $cmd .= ")\n";
}

# end of exp_ind_part_g

sub process_ind_part_ts
{
    # is ind_<name>_part_ts is in the form of XXXX#, expand it
    # else treat it as a comma separated list of ts names
    if (defined $params{'ind'.'.index.'_part_ts'})
    {
        @ind_part_ts = split(/,/, $params{'ind'.'.index.'_part_ts'});
        if ($ind_part_ts[0] =~ /\#/)
        {
            $filenm = shift(@ind_part_ts);
            $savename = $filenm;
            $params{'ind'.'.index.'_part_ts'} = "";
            for ($i = 1; $i <= $numpart; $i++)
            {
                $filenm =~ s/\#/$i/g;
                ($i == $numpart) ? ($addcomma = "") : ($addcomma = "," );
                $params{'ind'.'.index.'_part_ts'} =
                $params{'ind'.'.index.'_part_ts'} . $filenm . $addcomma;
                $filenm = $savename;
            }
            @ind_part_ts = split(/,/, $params{'ind'.'.index.'_part_ts'});
        }
        else {
            if (@ind_part_ts != $numpart)
            {
                $numpart = @ind_part_ts;
                if (($numpart % $numfil) == 0) {
                    $p = $numpart / $numfil;
                    $fil = "";
                    for ($i = 0; $i < $p; $i++)
                    {
                        if ($i == 0) {
                            $fil = $params{'ind'.'.index.'_part_ts'};
                        }
                        else {
                            $fil = join(',', $fil, $params{'ind'.'.index.'_part_ts'});
                        }
                    }
                    @ind_part_ts = split(/,/, $fil);
                }
                else {
                    print "Number of partitions $numpart for table $index\n doesn't match
                    ind_$index_part_ts parameter.\n";
                    exit(-1);
                }
            }
        }
    }
    }

    # end of process_ind_part_ts

    sub process_ind_part_brys
    {
        $cnt = 0;
        @ind_part_vals = ();

        foreach $col (@pcollist)
        {
            # add quotes for character strings and dates

            if (($params{$sob_stype.'_'.'.indtab.'_'.'.col.'_type'} ==
            /[cC][hH][aA][rR]/)) {
                ($params{$sob_stype.'_'.'.indtab.'_'.'.col.'_type'} ==
                /[dD][aA][tT][eE]/))
                {
                    $addquote = "\"";
                }

                @ind_part_col = ();

                # calculate the values for l_orderkey

                if ($col =~ /^l_orderkey$/)
                {
                    $high_l_orderkey = $params{'scale_factor'} * 1500000 * 4;
                    $pval = 1;
                    if (defined $params{$sob_stype.'_'.'.table.'_#part'})
                    {
                        $numpart = $params{$sob_stype.'_'.'.table.'_#part'};
                    }
                    else
                    {
                        $numpart = $params{'ind'.'.index.'_#part'};
                    }
                    $incrc = $high_l_orderkey / $numpart;
                    for ($i=0; $i < $numpart-1; $i++) {
                        $pval = $pval + $incrc;
                        push(@ind_part_col, $pval);
                    }
                    push(@ind_part_col, 'MAXVALUE');
                }
                else {
                    if ($col =~ /^l_partkey$/) {
                        $high_l_partkey = $params{'scale_factor'} * 200000;
                        $pval = 1;
                        if (defined $params{$sob_stype.'_'.'.table.'_#part'})
                        {
                            $numpart = $params{$sob_stype.'_'.'.table.'_#part'};
                        }
                        else
                        {
                            $numpart = $params{'ind'.'.index.'_#part'};
                        }
                        $incrc = $high_l_partkey / $numpart;
                        for ($i=0; $i < $numpart-1; $i++) {
                            $pval = $pval + $incrc;
                            push(@ind_part_col, $pval);
                        }
                        push(@ind_part_col, 'MAXVALUE');
                    }
                    else
                    {
                        @ind_part_col =
                        split(/,/, $params{'ind'.'.index.'_'.'.col.'_partvals'});
                    }
                }

                if (@ind_part_col != $params{'ind'.'.index.'_#part'})
                {
                    print "Number of partition boundary values $d for column $col in
                    global index $index doesn't match the number of partitions
                    ($params{'ind'.'.index.'_#part'}) of the index\n", ($ind_part_col + 1);
                    exit(-1);
                }
                for ($i=0; $i < $params{'ind'.'.index.'_#part'}; $i++)
                {
                    ($cnt == $#pcollist) ? ($addcomma = "") : ($addcomma = "," );
                    if ($ind_part_col[$i] =~ /[Mm][Aa][Xx][Vv][Aa][Ll][Uu][Ee]/) {
                        $addquote = "\"";
                    }
                    if ($params{$sob_stype.'_'.'.indtab.'_'.'.col.'_type'} == /[dD][aA][tT][eE]/)
                    {
                        if ($i == $params{'ind'.'.index.'_#part'} - 1)
                        {
                            $ind_part_vals[$i] = "MAXVALUE";
                        }
                        else {
                            $nls_format = (defined
                            $params{$sob_stype.'_'.'.indtab.'_'.'.col.'_date_format'}) ? "YYYY-MM-DD";
                            $params{$sob_stype.'_'.'.indtab.'_'.'.col.'_date_format'} : "YYYY-MM-DD";
                            $ind_part_vals[$i] = "to_date('". $ind_part_col[$i] . "','";
                            $ind_part_vals[$i] = "nls_format.'"')". $addcomma;
                        }
                    }
                    elseif (!($params{$sob_stype.'_'.'.indtab.'_'.'.col.'_type'} ==
                    /[iI][nN][tT][eE][gG][eE][rR]/))
                    {
                        ($params{$sob_stype.'_'.'.indtab.'_'.'.col.'_type'} == /[nN][uU][mM][bB][eE][rR]/))
                        {
                            if ($index =~ /l_ored/) {
                                print "Before:$ind_part_vals[$i]\n";
                            }
                            $ind_part_vals[$i] = $ind_part_vals[$i] . $addquote .
                            $ind_part_col[$i] . $addquote . $addcomma;
                            if ($index =~ /l_ored/) {
                                print "After:$ind_part_vals[$i]\n";
                            }
                        }
                    }
                }
            }
        }
    }
}

```

```

    } else {
        $ind_part_vals[$i] = $ind_part_col[$i]. $addcomma ;
    }
    $cnt++;
} # end of process_ind_part_brys

sub analyz
{
    &dump0("#####");
    &dump0("# Analyze Phase");
    &time0("Begin analyze");

    if ($params{'analyze_type'} == /via package dbms.stats/)
    {
        &time0("Begin via package dbms.stats analyzing");
        foreach $object (@analyzlist)
        {
            $cmd = "connect $params{'user'}/$params{'passwd'};\n";
            $phead="anl_.$object;
            $type = $params{$phead.'_object_type'};
            $type = "table" if ($params{$phead.'_object_type'} == /view/);
            $type = "table" if ($params{$phead.'_object_type'} == /viewlog/);
            $objectname = $object;
            $objectname = "MLOG\\\\"$object if ($params{$phead.'_object_type'} == /viewlog/);
            $cmd .= "execute
            dbms_stats.$params{$phead.'_anl_type'}.stats('$params{'user'}';
            $cmd .= "estimate percent => $params{$phead.'_percent'}" if (defined
            $params{$phead.'_percent'});
            $cmd .= "degree => $params{$phead.'_degree'}" if (defined
            $params{$phead.'_degree'});
            $cmd .= "granularity => '$params{$phead.'_granularity'}'" if (defined
            $params{$phead.'_granularity'});
            $cmd .= "block_sample => $params{$phead.'_block_sample'}" if (defined
            $params{$phead.'_block_sample'});
            $cmd .= " '$objectname.'" if !(($object == /schema/));
            $cmd .= ";\n";
            &dumpl("sql");
            &dump0("wait") if ($object == /schema/);
        }
        &time0("End per schema analyzing");
    }
    else
    {
        foreach $object (@analyzlist)
        {
            &time0("Begin analyzing $object");
            &advmulti();
            if (($params{'anl_.$object._type'} == /table/) &&
            ($params{'tab_.$object._#part'} > 1)) {
                $tab_entry = 'tab_.$object;
                if (!defined $params{$tab_entry.'_partnames'})
                {
                    # if no partnames are specified, use a default part name
                    for ($i = 1; $i <= $params{$tab_entry.'_part'}; $i++)
                    {
                        ($i == $params{$tab_entry.'_part'}) ? ($addcomma = ",") :
                        ($addcomma = ",");
                        $nextfile = sprintf("%s%s%s", $table, "_p",
                        $i, $addcomma);
                        $params{$tab_entry.'_partnames'} =
                        $params{$tab_entry.'_partnames'} . $nextfile;
                    }
                }
                @tab_partnames = split(/,/, $params{$tab_entry.'_partnames'});
                # if partnames is specified as XXXX#, then expand
                if ($tab_partnames[0] == /\#/)
                {
                    $filenm = shift(@tab_partnames);
                    $savename = $filenm;
                    $params{$tab_entry.'_partnames'} = "";
                    for ($i = 1; $i <= $params{$tab_entry.'_part'}; $i++)
                    {
                        $filenm = shift(@tab_partnames);
                        ($i == $params{$tab_entry.'_part'}) ? ($addcomma = ",") :
                        ($addcomma = ",");
                        $params{$tab_entry.'_partnames'} =
                        $params{$tab_entry.'_partnames'} . $filenm . $addcomma;
                        $filenm = $savename;
                    }
                    @tab_partnames = split(/,/, $params{$tab_entry.'_partnames'});
                    printf("Expanded $savename to:\n$params{$tab_entry.'_partnames'}\n\n") if
                    $verbose;
                }
                else {
                    if (@tab_partnames != $params{$tab_entry.'_part'})
                    {
                        print "Number of partitions $params{$tab_entry.'_part'} for $table doesn't
                        match\n_partnames parameter for $params{$tab_entry.'_partnames'}\n";
                        exit(-1);
                    }
                }
                @tab_partnames = split(/,/, $params{'tab_.$object._partnames'});
                foreach $partname (@tab_partnames)
                {
                    $cmd = "connect $params{'user'}/$params{'passwd'};\n";
                    $cmd .= "analyze $params{'anl_.$object._type'} $object ";
                    $cmd .= "partition ($partname) ";
                    if (defined $params{'anl_.$object._estimate'})
                    {
                        $cmd .= "estimate statistics $params{'anl_.$object._estimate'};\n";
                    }
                    else
                    {
                        $cmd .= "compute statistics;\n";
                    }
                    &dumpl("sql");
                }
            }
            &time0("Done analyzing $object");
        }
        &time0("End analyze");
    }
}

sub chdop_old
{
    &dump0("#####");
    &dump0("# Make DOP Phase");

    if (@chdoplist) {
        $cmd = "connect $params{'user'}/$params{'passwd'};\n";
        foreach $object (@chdoplist)
        {
            if ($params{'chdop_.$object._type'} == /table/) {
                $cmd .= "alter table $object parallel (degree $chdoplist{$object}
                instances 1);\n";
            }
            else {
                $cmd .= "alter index $object parallel (degree $chdoplist{$object}
                instances 1);\n";
            }
        }
        foreach $object (@chstorlist)
        {
            if ($params{'chstor_.$object._type'} == /table/) {

```



```

        last WTLOOP;
    }
}

}
}
}
if ($params{'qry_separate_queries'} == /true/)
{
    &dump2("sql2");
}
}
if ($params{'qry_separate_queries'} != /true/)
{
    &dump2("sql2");
}
}
}
}

#*****
#*****
# Utility Functions
#*****
#*****

sub prepmulti
{
    # Some multi-instance stuff
    $num_inst = $params{'phase.'$_num_inst'};
    if ($num_inst > 1)
    {
        $multi = 1;
        $cur_inst = 1;
        &advmulti(); # Make sure it's set to some appropriate value
    }
    else
    {
        $multi = 0;
    }
}

sub advmulti
{
    if ($multi)
    {
        &setinst($cur_inst);
        $cur_inst = ($cur_inst % $num_inst) + 1;
    }
}

sub startdb
{
    # Get init.oras for startup/shutdown
    $checkios = 0;
    $ifile = sprintf("%s%s", $params{'dbs_area'},
        "init_${params{'oracle_sid'}.ora}");
    $checkios = 1 if (!-e $ifile);
    for ($j = 1; $j <= $num_inst; $j++)
    {
        $iofname[$j] = sprintf("%s%s", $params{'dbs_area'},
            "init" . $j . "_" . $params{'oracle_sid'}.ora);
        $iofname[$j] = sprintf("/host%s", $iofname[$j])
            if ($params{'special_machine'} eq 'ncube');
        $iofname[$j] =~ s/\?/$ENV{'ORACLE_HOME'}/g;
        $checkios = 1 if (!-e $iofname[$j]);
    }
    &createios() if $checkios;

    # startup exclusive/shared single-instance/multi-instance
    if ($multi)
    {
        for ($loop = 1; $loop <= $num_inst; $loop++)
        {
            &setinst($loop);
            $cmd = "startup shared pfile=$iofname[$loop];\n";
            &dump1("sql");
            &dump0("wait") if ($params{'sync_startup'} == /true/);
        }
    }
    else
    {
        $cmd = "startup pfile=$iofname[1];\n";
        &dump1("sql");
    }
    &dump0("wait");
}

sub shutdb
{
    # shutdown whatever might be running
    if ($multi)
    {
        for ($loop = $num_inst; $loop >= 1; $loop--)
        {
            &setinst($loop);
            $cmd = "shutdown ${params{'shutdown_level'}};\n";
            &dump1("sql");
        }
    }
    else
    {
        $cmd = "shutdown ${params{'shutdown_level'}};\n";
        &dump1("sql");
    }
    &dump0("wait");
}

sub create_ts
{
    $tsname = shift(@_); # 1. parameter is the list of tablespaces

    &dump0("# creating ${params{'user'}}'s $tsname tablespace");
    &advmulti();

    @ts_datafiles = split(/,/, $params{$tsname.'_datafiles'});
    $ts_datafile = shift (@ts_datafiles);

    if ($params{$tsname.'_managed'} == /uU[u] [sS] [eE] [rR]/)
    {
        # user managed temporary tablespace
        if ($params{$tsname.'_temporary'} == /tT[t] [rR] [uU] [eE]/)
        {
            # temporary tablespace
            $cmd .= "drop tablespace $tsname;\n";
            $cmd .= "create tablespace $tsname temporary;\n";
        }
        else # user managed permanent tablespace
        {
            $cmd .= "drop tablespace $tsname;\n";
            $cmd .= "create tablespace $tsname;\n";
        }
    }
    $cmd .= "datafile '${params{$tsname.'_area'}}$ts_datafile' size
        ${params{$tsname.'_first_size'}} ${params{$tsname.'_options'}}\n";
    $cmd .= "default storage ${params{$tsname.'_storage'}}\n" if defined
        ${params{$tsname.'_storage'}};
    }
    else # system managed temporary tablespace
    {
        if ($params{$tsname.'_temporary'} == /tT[t] [rR] [uU] [eE]/)
        {
            # temporary tablespace
            $cmd .= "drop tablespace $tsname;\n";
            $cmd .= "create temporary tablespace $tsname;\n";
            $cmd .= "tempfile '${params{$tsname.'_area'}}$ts_datafile' size
                ${params{$tsname.'_first_size'}} ${params{$tsname.'_options'}}\n";
            $cmd .= "extent management local;\n";
            if (defined ($params{$tsname.'_extent_size'})) && ($params{$tsname.'_extent_size'}
                == /[aA] [uU] [tT] [oO] [aA] [lL] [lL] [oO] [cC] [aA] [tT] [eE]/))
            {
                $cmd .= "autoallocate;\n";
            }
            elsif (defined ($params{$tsname.'_extent_size'})) &&
                !($params{$tsname.'_extent_size'} ==
                    /[aA] [uU] [tT] [oO] [aA] [lL] [lL] [oO] [cC] [aA] [tT] [eE]/))
            {
                $cmd .= "uniform size " . $params{$tsname.'_extent_size'} . "\n";
            }
        }
        else # system managed permanent tablespace
        {
            $cmd .= "drop tablespace $tsname;\n";
            $cmd .= "create tablespace $tsname;\n";
            $cmd .= "datafile '${params{$tsname.'_area'}}$ts_datafile' size
                ${params{$tsname.'_first_size'}} ${params{$tsname.'_options'}}\n";
            $cmd .= "extent management local;\n";
            if (defined ($params{$tsname.'_extent_size'})) &&
                ($params{$tsname.'_extent_size'} ==
                    /[aA] [uU] [tT] [oO] [aA] [lL] [lL] [oO] [cC] [aA] [tT] [eE]/))
            {
                $cmd .= "autoallocate;\n";
            }
            elsif (defined ($params{$tsname.'_extent_size'})) &&
                !($params{$tsname.'_extent_size'} ==
                    /[aA] [uU] [tT] [oO] [aA] [lL] [lL] [oO] [cC] [aA] [tT] [eE]/))
            {
                $cmd .= "uniform size " . $params{$tsname.'_extent_size'} . "\n";
            }
        }
        else
        {
            $cmd .= "uniform";
        }
    }
    $cmd .= "nologging\n" if ($params{$tsname.'_nolg'} == /tT[t] [rR] [uU] [eE]/);
    $cmd .= ";\n";
    &dump1("sql");
}

sub add_dfs
{
    $tsname = shift(@_);
    &dump0("# adding ${params{'user'}}'s $tsname datafiles");
    @ts_datafiles = split(/,/, $params{$tsname.'_datafiles'});
    $ts_datafile = shift (@ts_datafiles); # drop first one
    $cur_inst = 2 if $multi; # for possible locality on SP2
    for ($i = 1; $i < $params{$tsname.'_files'}; $i++)
    {
        $ts_datafile = shift (@ts_datafiles);
        &advmulti();
        $cmd .= "alter tablespace $tsname\n";
        if ($params{$tsname.'_managed'} == /uU[u] [sS] [eE] [rR]/)
        {
            # user managed temporary tablespace
            $cmd .= " add datafile ";
        }
        else
        {
            # system managed temporary tablespace
            if ($params{$tsname.'_temporary'} == /true/)
            {
                $cmd .= " add tempfile ";
            }
            else
            {
                $cmd .= " add datafile ";
            }
        }
        $cmd .= "'${params{$tsname.'_area'}}$ts_datafile' size ${params{$tsname.'_size'}}
            ${params{$tsname.'_options'}};\n";
        &dump1("sql");
    }
}

sub createios
{
    push (@iokv, "db_name=${params{'oracle_sid'}}");
    push (@iokv, "control_files=${params{'io_control_files'}}");
    push (@iokv, "db_block_buffers=1000");
    push (@iokv, "shared_pool_size=350000000");
    push (@iokv, "parallel_max_servers=144");
}

```

```

push (@iokv, "parallel_min_servers=0");
push (@iokv, "max_dump_file_size=5000");
push (@iokv, "audit_trail=FALSE");
push (@iokv, "global_names=FALSE");
# push (@iokv, "commit_point_strength=1");
# push (@iokv, "dblink_encrypt_login=true");
# push (@iokv, "db_block_size=16384");
push (@iokv, "db_block_size=8192");
push (@iokv, "db_file_multiblock_read_count=32");
push (@iokv, "processes=256");
push (@iokv, "sessions=256");
push (@iokv, "transactions=512");
push (@iokv, "transactions_per_rollback_segment=20");
push (@iokv, "max_rollback_segments=256");
push (@iokv, "distributed_transactions=0");
push (@iokv, "nls_date_format=YYYY-MM-DD");
push (@iokv, "db_files=1023");
push (@iokv, "open_cursors=1024");
push (@iokv, "optimizer_mode=CHOOSE");
push (@iokv, "optimizer_percent_parallel=100");
push (@iokv, "sort_area_size=3000000");
push (@iokv, "always_semi_join=HASH");
push (@iokv, "parallel_broadcast_enabled=TRUE");
push (@iokv, "optimizer_features_enable=8.0.4");
push (@iokv, "compatible=8.0.4");
push (@iokv, "hash_multiblock_io_count=64");
push (@iokv, "always_anti_join=HASH");

$ifile = sprintf ("%s%s", $params{'dbs_area'},
"init_$params{'oracle_sid'}.ora");
$ifile -- s/\?/ENV{'ORACLE_HOME'}/g;
if ($params{'skip_mk_initoras'} != /include/)
{
    open (IOFILE, ">$ifile");
    print IOFILE "# Init.ora include file created by bumpx.pl\n\n";
    while ($kv = shift(@iokv))
    {
        $kv =~ /(.*)=(.*)/;
        $entry = sprintf ("%s-25s = %s\n", $1, $2);
        print IOFILE $entry;
    }
    if (defined $params{'io_file'})
    {
        $entry = sprintf ("%s-25s = %s\n", "ifile", $params{'io_file'});
        print IOFILE $entry;
    }
    close (IOFILE);

    for ($j = 1; $j <= $num_inst; $j++)
    {
        $iofname[$j] = sprintf ("%s%s", $params{'dbs_area'},
"init".$j."_$params{'oracle_sid'}.ora");
        $iofname_opn = $iofname[$j];
        $iofname_opn -- s/\?/ENV{'ORACLE_HOME'}/g;
        if ($params{'skip_mk_initoras'} != /instance/)
        {
            open (IOFILE, ">$iofname_opn");
            print IOFILE "# Init.ora for instance $j created by bumpx.pl\n\n";
            print IOFILE "thread = $j\n";
            if ($params{'special_machine'} ne 'ncube')
            {
                print IOFILE "ifile = $ifile\n";
                close (IOFILE);
            }
            $iofname[$j] = sprintf ("/host%s", $iofname[$j])
            if ($params{'special_machine'} eq 'ncube');
        }
    }
}

sub create_objects
{
    $ob_type = shift(@_);
    @ob_objcstlist = @_;
    $cmd = "connect $params{'user'}/$params{'passwd'};\n";
    foreach $object (@ob_objcstlist)
    {
        &create_object;
    }
    &dumpl ("**sql");
}

sub create_object
# this routine only creates one object
# need it because of the creation of materialized views
{
    $ob_entry = $ob_type.'.'.$object;
    if (($ob_type cmp 'tab') == 0) # we are creating a table
    {
        $cmd = "drop table $object;\n";
        $cmd = "create table $object";
    }
    elsif (($ob_type cmp 'view') == 0) # we are creating a view
    {
        if ($params{$ob_entry.'_creation_type'} == /[dD][iI][rR][eE][cC][tT]/) # direct
        {
            $cmd = "drop materialized view $object;\n";
            $cmd = "create materialized view $object";
        }
        else # with table
        {
            $cmd = "drop table $object;\n";
            $cmd = "create table $object";
        }
        if (($ob_type cmp 'viewlog') == 0) # we have a viewlog
    }
    {
        $cmd = "drop materialized view log on $object;\n";
        $cmd = "create materialized view log ";
        $cmd = "on $object;\n";
    }
    elsif (($ob_type eq 'view') ||
    (($ob_type eq 'view') && ($params{$ob_entry.'_creation_type'} ==
/[wW][iI][tT][hH][sS][tT][aA][bB][iI][eE][tT]))
    {
        $cmd = "\n";
        $cmd = "$params{$ob_entry.'_cons'}\n" if defined $params{$ob_entry.'_cons'};
        @ob_collist = split(/,/, $params{$ob_entry.'_columns'});
        &add_columns; # the list of columns are in @ob_collist
        $cmd = "\n";
    }
    else
    {
        $cmd = "\n";
    }
    if (defined $params{$ob_entry.'_cluster'})
    {
        $cmd = "cluster $params{$ob_entry.'_cluster'} ($params{$ob_entry.'_clucols'})\n";
    }
    else
    {
        # add organization and pctthreshold for index only tables
        $cmd = "organization $params{$ob_entry.'_org'}\n" if defined
$params{$ob_entry.'_org'};
        $cmd = "pctthreshold $params{$ob_entry.'_pct'}\n" if defined
$params{$ob_entry.'_pct'};
        $cmd = "pctfree $params{$ob_entry.'_pctf'}\n" if defined $params{$ob_entry.'_pctf'};
        $cmd = "pctused $params{$ob_entry.'_pctu'}\n" if defined $params{$ob_entry.'_pctu'};
        $cmd = "initrans $params{$ob_entry.'_it'}\n" if defined $params{$ob_entry.'_it'};
        $cmd = "maxtrans $params{$ob_entry.'_mt'}\n" if defined $params{$ob_entry.'_mt'};
        $cmd = "tablespace $params{$ob_entry.'_ts'}\n" if defined
$params{$ob_entry.'_ts'};
        $cmd = "storage $params{$ob_entry.'_storage'}\n" if defined
$params{$ob_entry.'_storage'};
        if (defined $params{$ob_entry.'_of_ts'})
        {
            # overflow tablespace specs
            $cmd = "overflow ";
            $cmd = "pctfree $params{$ob_entry.'_of_pctf'} " if defined
$params{$ob_entry.'_of_pctf'};
            $cmd = "pctused $params{$ob_entry.'_of_pctu'} " if defined
$params{$ob_entry.'_of_pctu'};
            $cmd = "initrans $params{$ob_entry.'_of_it'} " if defined
$params{$ob_entry.'_of_it'};
            $cmd = "maxtrans $params{$ob_entry.'_of_mt'} " if defined
$params{$ob_entry.'_of_mt'};
            $cmd = "tablespace $params{$ob_entry.'_of_ts'} " if defined
$params{$ob_entry.'_of_ts'};
            $cmd = "storage $params{$ob_entry.'_of_storage'}\n" if defined
$params{$ob_entry.'_of_storage'};
        }
        if ((defined $params{$ob_entry.'_pardeg'}) || (defined
$params{$ob_entry.'_parinst'}))
        {
            $cmd = "parallel";
            if ((defined $params{$ob_entry.'_pardeg'}) &&
($params{$ob_entry.'_pardeg'} == /[dD][eE][fF][aA][uU][lL][tT]/))
            {
                $cmd = "\n";
            }
            else
            {
                $cmd = "degree $params{$ob_entry.'_pardeg'} " if defined
$params{$ob_entry.'_pardeg'};
                $cmd = "instances $params{$ob_entry.'_parinst'} " if defined
$params{$ob_entry.'_parinst'};
                $cmd = "\n";
            }
        }
        $cmd = "nologging\n" if ($params{$ob_entry.'_nolg'} == /[tT][rR][uU][eE]/);
        $cmd = "cache\n" if ($params{$ob_entry.'_cache'} == /[tT][rR][uU][eE]/);
    }
    # add partition support
    if (defined $params{$ob_entry.'_part'}) && ($params{$ob_entry.'_part'} > 1)
    {
        &expand_partitions ($ob_entry,$object,$ob_type);
        # no partitioning
    }
    if (((($ob_type cmp 'view') == 0) && ($params{$ob_entry.'_creation_type'} ==
/[dD][iI][rR][eE][cC][tT]/))
    {
        $cmd = "BUILD ".$params{$ob_entry.'_build_when'}." " if (defined
$params{$ob_entry.'_build_when'});
        &add_refresh_clause($ob_entry);
        if ((defined $params{$ob_entry.'_rewrite'}) && ($params{$ob_entry.'_rewrite'} ==
/[tT][rR][uU][eE]/))
        {
            $cmd = "enable query rewrite\n";
        }
        elsif ($params{$ob_entry.'_rewrite'} == /[fF][aA][lL][sS][eE]/)
        {
            $cmd = "disable query rewrite\n";
        }
    }
    if (defined $params{$ob_entry.'_as_select'}) # as select (here I only copy a
variable called <viewname>_define_as_select
    {
        &print_as_select_stm($params{$ob_entry.'_as_select'});
        $cmd = "\n";
    }
    if (($ob_type cmp 'viewlog') == 0)
    {
        $cmd = "with rowid ";
        if (defined $params{$ob_entry.'_columns'})
        {
            $cmd = "\n";
            @ob_collist = split(/,/, $params{$ob_entry.'_columns'});
            &add_columns; # the list of columns are in @ob_collist
        }
    }
}

```

```

$cmd .= "\n";
}
}

if ($params{$ob_entry.'creation_type'} ==
/[wW][iI][tT][hH][sS][tT][aA][tTbB][lL][eE]/)
{
$cmd .= ";ndrop materialized view $object;\n";
$cmd .= "create materialized view $object \n";
$cmd .= "BUILD ".$params{$ob_entry.'_build_when'}." \n" if (defined
$params{$ob_entry.'_build_when'});
$cmd .= "on prebuilt table \n";
#If (defined $params{$ob_entry.'_precision'})
#{
# ($params{$ob_entry.'_precision'} == /[rR][eE][dD][uU][cC][eE][dD]/) ?
# $cmd .= "WITH REDUCED PRECISION \n"
# : $cmd .= "WITHOUT REDUCED PRECISION \n";
#}

&add_refresh_clause($ob_entry);

if ((defined $params{$ob_entry.'rewrite'}) &&
($params{$ob_entry.'_rewrite'} =~ /[tT][rR][uU][eE]/))
{
$cmd .= "enable query rewrite\n";
}
elseif ($params{$ob_entry.'_rewrite'} == /[fF][aA][lL][sS][eE]/)
{
$cmd .= "disable query rewrite\n";
}

$cmd .= "refresh ".$params{$ob_entry.'_refresh'}." \n" if (defined
$params{$ob_entry.'_refresh'});
$cmd .= $params{$ob_entry.'_queryrw'}." \n" if (defined
$params{$ob_entry.'_queryrw'});
&print_as_select_stm($params{$ob_entry.'_as_select'});
}
$cmd .= "; \n";
}

sub add_columns
{
while ($col = shift(@ob_collist))
{
(@ob_collist == 0) ? ($addcomma = "") : ($addcomma = ",";);
if (($ob_type cmp 'tab') == 0)
{
$column_type = $params{$ob_entry.'_'.$col.'_type'};
}
else
{
$column_type = "";
}
$nextline = sprintf (" %20s %s %s$addcomma\n", $col, $column_type,
$params{$ob_entry.'_'.$col.'_cons'});
$cmd .= $nextline;
}
}

sub add_refresh_clause
{
$ob_entry = $_[0];
if (defined $params{$ob_entry.'_refresh_how'})
{
$cmd .= "REFRESH ".$params{$ob_entry.'_refresh_how'}." \n";
$cmd .= "ON ".$params{$ob_entry.'_refresh_when'}." \n" if (defined
$params{$ob_entry.'_refresh_when'});
}
}

sub print_as_select_stm
{
if ($_[0] =~ /\/*/)
{
($beforehint,$rest)=split(/\/*/,$_[0]);
($hint,$as_select) =split(/\*\/,$rest);
}
else
{
$as_select = $_[0];
$hint="";
}
$as_select =~ s/,/,,\n/g;
$as_select =~ s/from/\nFROM\n/g;
$as_select =~ s/select/SELECT\n/g;
$as_select =~ s/where/\nWHERE\n/g;
$as_select =~ s/group by/\nGROUP BY\n/g;
$as_select =~ s/ and /\n AND \n/g;
$as_select =~ s/ or /\n OR \n/g;
$cmd .= "as select\n";
$cmd .= "/*$hint.$as_select*/." \n" if ($hint =~ /\^/);
$cmd .= $as_select;
}

sub create_clusters
{
$cmd .= "connect $params{'user'}/$params{'passwd'};\n";
foreach $cluster (@clulist)
{
$cmd .= "drop cluster $cluster including tables;\n";
$cmd .= "create cluster $cluster (\n";
@clu_collist = split(/,/, $params{'clu_'.$cluster.'_columns'});
while ($col = shift(@clu_collist))
{
(@clu_collist == 0) ? ($addcomma = "") : ($addcomma = ",";);
$nextline = sprintf (" %20s %s$addcomma\n", $col,
$params{'clu_'.$cluster.'_'.$col.'_type'});
$cmd .= $nextline;
}
$cmd .= "\n";
$cmd .= "pctfree $params{'clu_'.$cluster.'_%f'}\n" if (defined
$params{'clu_'.$cluster.'_%f'});
}
}

$params{$cmd} = "pctused $params{'clu_'.$cluster.'_%u'}\n" if (defined
$params{'clu_'.$cluster.'_%u'});
$params{$cmd} = "initrans $params{'clu_'.$cluster.'_it'}\n" if (defined
$params{'clu_'.$cluster.'_it'});
$params{$cmd} = "maxtrans $params{'clu_'.$cluster.'_mt'}\n" if (defined
$params{'clu_'.$cluster.'_mt'});
$params{$cmd} = "size $params{'clu_'.$cluster.'_size'}\n" if (defined
$params{'clu_'.$cluster.'_size'});
$params{$cmd} = "tablespace $params{'clu_'.$cluster.'_ts'}\n" if (defined
$params{'clu_'.$cluster.'_ts'});
$params{$cmd} = "storage $params{'clu_'.$cluster.'_storage'}\n" if (defined
$params{'clu_'.$cluster.'_storage'});
$cmd .= "index\n" if ($params{'clu_'.$cluster.'_index'} == true);
if (defined $params{'clu_'.$cluster.'_hashkeys'})
{
$cmd = "hash is $params{'clu_'.$cluster.'_hashcol'} " if (defined
$params{'clu_'.$cluster.'_hashcol'});
$cmd = "hashkeys $params{'clu_'.$cluster.'_hashkeys'}\n";
}
if ((defined $params{'clu_'.$cluster.'_parinst'}) || (defined
$params{'clu_'.$cluster.'_parinst'}))
{
$cmd = "parallel ";
$cmd = "degree $params{'clu_'.$cluster.'_pardeg'} " if (defined
$params{'clu_'.$cluster.'_pardeg'});
$cmd = "instances $params{'clu_'.$cluster.'_parinst'}" if (defined
$params{'clu_'.$cluster.'_parinst'});
$cmd = "\n";
}
$cmd = "cache\n" if $params{'clu_'.$cluster.'_cache'} == true/;
$cmd = ".\n";
# Now, create the cluster index, if necessary
if ($params{'clu_'.$cluster.'_index'} == true)
{
$cluidx = "ind_".$cluster;
$cmd = "drop index $cluidx;\n";
$cmd = "create index $cluidx on cluster $cluster;\n";
}
&dumple("sql");
}

sub expand_partitions
{
# *MP* parameterize procedure: changed $tab_entry into $ob_entry 1. parameter
$_[0]
# $_[1] stable into $ob 2. parameter
# $_[2] introduced $ob_type 3. parameter
{
$ob_entry = $_[0];
$ob = $_[1];
$ob_type = $_[2];

if ($params{$ob_entry.'_parttype'} == /[rR][aA][nN][gG][eE]/)
{
$cmd = "partition by range (" . $params{$ob_entry.'_partcol'} . ") \n";
&expand_partitions_doit($ob_entry,$ob,$ob_type);
}
elseif ($params{$ob_entry.'_parttype'} == /[hH][aA][sS][hH]/)
{
$cmd = "partition by hash (" . $params{$ob_entry.'_partcol'} . ") \n";
$cmd = "partitions " . $params{$ob_entry.'_#part'} . " \n";
}
elseif ($params{$ob_entry.'_parttype'} ==
/[cC][oO][mM][pP][oO][sS][iI][tT][eE]/)
{
$cmd = "partition by range (" . $params{$ob_entry.'_partcol'} . ") \n";
$cmd = "subpartition by hash(" . $params{$ob_entry.'_subpartcol'} . ") \n";
$cmd = "subpartitions " . $params{$ob_entry.'_#subpart'} . " \n";
&expand_partitions_doit($ob_entry,$ob,$ob_type);
}
}# no partitioning

sub expand_partitions_doit
{
# *MP* parameterize procedure: changed $tab_entry into $ob_entry 1. parameter
$_[0]
# $_[1] stable into $ob 2. parameter
# $_[2] introduced $ob_type 3. parameter
{
$ob_entry = $_[0];
$ob = $_[1];
$ob_type = $_[2];

@pcollist = split(/,/, $params{$ob_entry.'_partcol'});
if (!defined $params{$ob_entry.'_partnames'})
{
# if no partnames are specified, use a default part name
for ($i = 1; $i <= $params{$ob_entry.'_#part'}; $i++)
{
($i == $params{$ob_entry.'_#part'}) ? ($addcomma = "") :
($addcomma = ",";);
$nextfile = sprintf("%$i$ob_p", $ob, $i, $addcomma);
$params{$ob_entry.'_partnames'} =
$params{$ob_entry.'_partnames'} . $nextfile;
}
}
$ob_partnames = split(/,/, $params{$ob_entry.'_partnames'});

# if partnames is specified as XXXX#, then expand
if ($ob_partnames[0] =~ /\#/)
{
$filenm = shift(@ob_partnames);
$savename = $filenm;
$params{$ob_entry.'_partnames'} = "";
for ($i = 1; $i <= $params{$ob_entry.'_#part'}; $i++)
{

```

```

$filenm =~ s/\#/$i/g;
($i == $params{$sob_entry.'_#part'}) ? ($addcomma = "") :
($addcomma = ",";);
$params{$sob_entry.'_partnames'} =
$params{$sob_entry.'_partnames'} . $filenm . $addcomma;
$filenm = $savename;
}
@ob_partnames = split(/,/, $params{$sob_entry.'_partnames'});
printf("Expanded $savename to:\n$params{$sob_entry.'_partnames'}\n\n") if $verbose;
} else {
if (@ob_partnames != $params{$sob_entry.'_#part'})
{
print "Number of partitions $params{$sob_entry.'_#part'} for $sob doesn't
match\n_partnames parameter for $params{$sob_entry.'_partnames'}\n";
exit(-1);
}
}
# now process the partition boundary
$process_part_brys ($sob_entry,$sob);
$process_part_ts ($sob_entry,$sob);

# complete the partition statement
for ($i=0; $i < $params{$sob_entry.'_#part'}; $i++)
{
$cmd .= "partition " . $sob_partnames[$i] . " values less than ";
if ($i==$params{$sob_entry.'_#part'}-1) {
$cmd .= "(MAXVALUE)\n";
}
else {
$cmd .= "(" . $sob_part_vals[$i] . ")\n";
}
$process_part_params ($sob_type, '$f', 'pctfree', $sob,$sob_partnames[$i]);
$process_part_params ($sob_type, 'it', 'initrans', $sob,$sob_partnames[$i]);
$process_part_params ($sob_type, 'mt', 'maxtrans', $sob,$sob_partnames[$i]);

if (defined $params{$sob_entry.'_' . $sob_partnames[$i].'_ts'})
{
$cmd .= 'tablespace ' . $params{$sob_entry.'_' . $sob_partnames[$i].'_ts'} . "\n";
$process_part_storage ($sob_type,$sob,$sob_partnames[$i]);
}
elseif (defined $params{$sob_entry.'_part_ts'}) {
$cmd .= 'tablespace ' . $sob_part_ts[$i] . "\n";
$process_part_storage ($sob_type,$sob,$sob_partnames[$i]);
}
}
if (defined $params{$sob_entry.'_' . $sob_partnames[$i].'_nolg'})
{
$cmd .= "nologging\n" if ($params{$sob_entry.'_' . $sob_partnames[$i].'_nolg'} ==
/[tT][rR][uU][eE]/);
}
elseif (defined $params{$sob_entry.'_part_def_nolg'}) {
$cmd .= "nologging\n" if ($params{$sob_entry.'_part_def_nolg'} ==
/[tT][rR][uU][eE]/);
}
$cmd .= (($i+1) == $params{$sob_entry.'_#part'}) ? "" : ",\n";
}
$cmd .= ")\n";
} #end expand_partitions_doit

sub process_part_storage
{
local ($type,$name,$pname) = @_;
if (defined $params{$type.'_' . $name.'_' . $pname.'_storage'})
{
$cmd .= 'storage ' . $params{$type.'_' . $name.'_' . $pname.'_storage'} . "\n";
}
elseif (defined $params{$type.'_' . $name.'_part_def_storage'}) {
$cmd .= 'storage ' . $params{$type.'_' . $name.'_part_def_storage'} . "\n";
}
}

sub process_part_params
{
local ($type,$p1,$p2,$name,$pname) = @_;
if (defined $params{$type.'_' . $name.'_' . $pname.'_' . $p1}) {
$cmd .= $p2 . " " . $params{$type.'_' . $name.'_' . $pname.'_' . $p1} . "\n";
}
elseif (defined $params{$type.'_' . $name.'_part_def_' . $p1}) {
$cmd .= $p2 . " " . $params{$type.'_' . $name.'_part_def_' . $p1} . "\n";
}
}

sub process_part_ts
{
# *MP* parameterize procedure: changed 'tab'.stable into $sob_entry 1. parameter
$_[0] (same as $sob_entry in expand_partition)
#
$_[1] (same as $sob in expand_partition) stable into $sob_ 2. parameter
{
$sob_entry = $_[0];
$sob_ = $_[1];

# is objecttype_<name>_part_ts in the form of XXXX#, expand it
# else treat it as a comma separated list of ts names
if (defined $params{$sob_entry.'_part_ts'})
{
@ob_part_ts = split(/,/, $params{$sob_entry.'_part_ts'});
if (@ob_part_ts[0] =~ /\#/)
{
$filenm = shift(@ob_part_ts);
$savename = $filenm;
$params{$sob_entry.'_part_ts'} = "";
for ($i = 1; $i <= $params{$sob_entry.'_#part'}; $i++)
{
$filenm =~ s/\#/$i/g;
($i == $params{$sob_entry.'_#part'}) ? ($addcomma = "") :
($addcomma = ",";);
$params{$sob_entry.'_part_ts'} =
$params{$sob_entry.'_part_ts'} . $filenm . $addcomma;
$filenm = $savename;
}
@ob_part_ts = split(/,/, $params{$sob_entry.'_part_ts'});
}
else {
if (@ob_part_ts != $params{$sob_entry.'_#part'})
{
if ($sob_part_ts != $params{$sob_entry.'_#part'})
{
$numfil = @ob_part_ts;
$sp = $params{$sob_entry.'_#part'} / $numfil;
$fil = "";
for ($i = 0; $i < $sp; $i++)
{
if ($i == 0) {
$fil = $params{$sob_entry.'_part_ts'};
}
else {
$fil = join(' ', $fil, $params{$sob_entry.'_part_ts'});
}
}
@ob_part_ts = split(/,/, $fil);
}
else {
print "Number of partitions $params{$sob_entry.'_#part'} for object $sob_\n
doesn't match $stable_part_ts parameter.\n";
exit (-1);
}
}
}
} # end of process_part_ts

sub process_part_brys

# *MP* parameterize procedure: changed 'tab'.stable into $sob_entry 1. parameter
$_[0] (same as $sob_entry in expand_partition)
#
$_[1] (same as $sob in expand_partition) stable into $sob_ 2. parameter
{
{
$sob_entry = $_[0];
$sob_ = $_[1];

$cnt = 0;
@ob_part_vals = ();

foreach $col (@pcollist)
{
# add quotes for character strings and dates

if (($params{$sob_entry.'_' . $col.'_type'} =~ /[cC][hH][aA][rR]/) ||
($params{$sob_entry.'_' . $col.'_type'} =~ /[dD][aA][tT][eE]/))
{
$addquote = "\"";
}
else {
$addquote = "";
}

@ob_part_col = split(/,/, $params{$sob_entry.'_' . $col.'_partvals'});
if (@ob_part_col != $params{$sob_entry.'_#part'})
{
printf "Number of partition boundary values %d for column $col in object $sob_
doesn't match the number of partitions ($params{$sob_entry.'_#part'}) of the
object\n", ($#ob_part_col + 1);
exit (-1);
}
for ($i=0; $i < $params{$sob_entry.'_#part'}; $i++)
{
($cnt == $#pcollist) ? ($addcomma = "") : ($addcomma = ",";);
if ($ob_part_col[$i] =~ /[Mm][Aa][Xx][Vv][Aa][Ll][Uu][Ee]/) {
$addquote = "\"";
}
if ($params{$sob_entry.'_' . $col.'_type'} =~ /[dD][aA][tT][eE]/)
{
if ($i == $params{$sob_entry.'_#part'} - 1)
{
$sob_part_vals[$i] = "MAXVALUE";
}
else {
$nlformat = (defined $params{$sob_entry.'_' . $col.'_date_format'}) ?
$params{$sob_entry.'_' . $col.'_date_format'} : "YYYY-MM-DD";
$sob_part_vals[$i] = "to_date('". $sob_part_col[$i]."', '$nlformat')";
$sob_part_vals[$i] = $nlformat . $addcomma;
}
}
else {
$sob_part_vals[$i] = $sob_part_vals[$i] . $addquote;
$sob_part_col[$i] = $sob_part_vals[$i] . $addquote . $addcomma;
}
}
$cnt++;
}
} # end of process_part_brys

sub alttabs
{
$di = shift(@_);
$di =~ /\(d*\)_\(\d*\)/;
$cmd = "";
$cmd .= "connect $params{'user'}/$params{'passwd'};\n";
$params{'alter_tables'} = $params{'tab_tables'} if !defined
$params{'alter_tables'};
@atabs = split(/,/, $params{'alter_tables'});
foreach $atab (@atabs)
{
$cmd .= "alter table $atab parallel (degree $1 instances $2);\n";
}
}

sub setinst
{
$inum = shift(@_);
$dump0 ("**inst");
$dump0 ("**wait");
}

```

```

sub dump2
{
    $value = shift(@_);
    &dump0($value);
    &dump0(" ");
    print OUTFILE $cmd;
    &dump0(" ");
    &dump0(" ");
    print OUTFILE $cmd2;
    &dump0(" ");
    $cmd = "";
    $cmd2 = "";
    print "." if $verbose;
}

sub dump1
{
    $value = shift(@_);
    &dump0($value);
    &dump0(" ");
    print OUTFILE $cmd;
    &dump0(" ");
    $cmd = "";
    print "." if $verbose;
}

sub dump0
{
    $value = shift(@_);
    $value = $value . "\n" if ($value != /.*\n$/);
    print OUTFILE "$value";
}

sub recreate_drive_extended_part
{
    $cmd = "creapart -d PhysicalDrive" . $drivenum . "\n";
    &dump1("**sh");
    $cmd = "Deleted partitions on PhysicalDrive" . $drivenum;
    &time0($cmd);
    $cmd = "creapart -x PhysicalDrive" . $drivenum . "\n";
    &dump1("**sh");
    $cmd = "Created extended partition on PhysicalDrive" . $drivenum;
    &time0($cmd);
}

sub create_drive_part
{
    $numfiles = $params{$ts_entry.'_#files'}-1;
    if ($d == 0)
    {
        $size = (defined $params{$ts_entry.'_first_size'}) ?
        $params{$ts_entry.'_first_size'} :
        $params{$ts_entry.'_size'};
    }
    else
    {
        $size = $params{$ts_entry.'_size'};
    }
    $size =~ s/[Mm]*/g;
    @files = split(/,/, $params{$ts_entry.'_datafiles'});
    foreach $file (@files)
    {
        &create_drive_part_file;
    }
}

sub create_drive_part_file
{
    # add 1MB to nt partition size to prevent writing data to cyl 0
    $psize = $size+1;
    $drivenum = $file =~ /^log/ ? $params{'plcre_log_drivenum'} :
    @drivenum[$d];
    $cmd = "creapart -l PhysicalDrive";
    $cmd .= $drivenum . " " . $psize . "\n";
    &dump1("**sh");
    $cmd = $file . "\tdevice\\PhysicalDrive" . $drivenum;
    $cmd .= "\\partition" . ++$partnum[$drivenum] . "\n";
    &time0($cmd);
    print LNKSFILE $cmd;
    if ($file =~ /^sys/ || $file =~ /^cntr/)
    {
        $d = $d < $md ? ++$d : 0;
    }
    elsif ($file =~ !/^log/)
    {
        $d = ($d < $md) && ($d < $numfiles) ? ++$d : 0;
    }
}

sub load_ctl_head
{
    open (CTLFILE, ">$control");
    print CTLFILE "----\n";
    print CTLFILE "---- $table.ctl for delimited records\n" if ($sud == 1);
    print CTLFILE "---- $table.ctl for fixed-length fields\n" if ($sud == 0);
    print CTLFILE "----\n\n";
    if (!($pre72) && defined $params{'tab_'. $table.'_loadextent'})
    {
        print CTLFILE "options\n";
        print CTLFILE "\n";
        print CTLFILE "storage = (initial $params{'tab_'. $table.'_loadextent'} next
        $params{'tab_'. $table.'_loadextent'})\n";
        print CTLFILE "\n";
    }
    print CTLFILE "unrecoverable\n" if ($params{'load_unrecoverable'} ==
    /[tT][rR][uU][eE]/);
    print CTLFILE "load\n";
    print CTLFILE "-- This is where INFILE should go.\n";
}

sub load_ctl_tail
{
    {
        print CTLFILE "$params{'load_insert_type'}\n";
        print CTLFILE "fields terminated by $params{'load_field_terminator'}\n" if
        ($sud == 1);
        print CTLFILE "\n";
        @tab_collist = split(/,/, $params{'tab_'. $table.'_columns'});
        while ($col = shift(@tab_collist))
        {
            (@tab_collist == 0) ? ($addcomma = "") : ($addcomma = ",");
            $pos = "";
            $pos = sprintf ("position (%s)", $params{'tab_'. $table.'_' . $col.'_pos'}) if ($sud ==
            0);
            $ctlline = sprintf (" %20s %s %s$addcomma", $col, $pos,
            $params{'tab_'. $table.'_' . $col.'_loadcolx'});
            print CTLFILE "$ctlline\n";
        }
        print CTLFILE "\n";
        close (CTLFILE);
    }

    #*****
    # Assigning Defaults
    #*****

    sub defaults
    {
        # defaults for the params associative array
        if (defined $bmttype)
        {
            eval(&$bmttype);
            &assigndefs;
        }
        &auxdefaults;
        &assigndefs;

        if (defined $params{'compatible'})
        {
            $pre72 = 1 if $params{'compatible'} =~ /7.0[1]/;
            $pre73 = 1 if $params{'compatible'} =~ /7.0[12]/;
        }

        foreach $phase (@phases)
        {
            $params{$phase.'_num_inst'} = $params{'def_num_inst'} if !defined
            $params{$phase.'_max_bg'} = $params{'max_bg'} if !defined
            $params{$phase.'_max_bg'};
            foreach $phase (@phases)
            {
                $params{'startupfile_'. $phase} =
                $params{'dbs_area'}. $params{'startupfile_'. $phase};
                $params{'startupfile_'. $phase} = sprintf ("/host%s",
                $params{'startupfile_'. $phase});
                if ($params{'special_machine'} eq 'ncube');
                {
                    $params{'startupfile_'. $phase} =~ s/\?/ENV{'ORACLE_HOME'}/g;
                }
                $params{'ts_undo_#rs'} = $params{'def_num_inst'} if !defined
                $params{'ts_undo_#rs'};
                $params{'ts_log_#files'} = $params{'ts_log_#threads'} *
                $params{'ts_log_#files_pt'};

                $params{'dbs_area'} = s/\?/ENV{'ORACLE_HOME'}/g;
                $params{'ts_def_area'} = $params{'dbs_area'} if !defined
                $params{'ts_def_area'};
                $params{'load_flatfile_area'} = $params{'ts_def_area'} if !defined
                $params{'load_flatfile_area'};
                $params{'load_controlfile_area'} = $params{'ts_def_area'} if !defined
                $params{'load_controlfile_area'};
                $params{'load_otherfile_area'} = $params{'ts_def_area'} if !defined
                $params{'load_otherfile_area'};

                $params{'ts_index_names'} = "ts_index" if (!defined
                $params{'ts_index_names'}) && ($params{'skip_ts'} != /index/);
                $params{'ts_temp_names'} = "ts_temp" if (!defined $params{'ts_temp_names'})
                && ($params{'skip_ts'} != /temp/);
                $params{'ts_data_names'} = "ts_data" if (!defined $params{'ts_data_names'})
                && ($params{'skip_ts'} != /data/);

                # expand ts_data groups

                if (defined $params{'ts_data_groups'})
                {
                    @ts_data_group = split(/,/, $params{'ts_data_groups'});
                    foreach $gname (@ts_data_group)
                    {
                        $nts = $params{$gname.'_group_ts'};
                        for ($i=0; $i<$nts; $i++)
                        {
                            $tsn = sprintf("%s%d", $gname, ($i+1));
                            $params{'ts_data_names'} .= ", " . $tsn;
                            $params{$tsn.'_datafiles'} = $tsn . ".#";
                            $params{$tsn.'_#files'} = $params{$gname.'_group_files'} if defined
                            $params{$gname.'_group_files'};
                            $params{$tsn.'_option'} = $params{$gname.'_group_option'} if defined
                            $params{$gname.'_group_option'};
                            $params{$tsn.'_area'} = $params{$gname.'_group_area'} if defined
                            $params{$gname.'_group_area'};
                            $params{$tsn.'_storage'} = $params{$gname.'_group_storage'} if defined
                            $params{$gname.'_group_storage'};
                            $params{$tsn.'_first_size'} = $params{$gname.'_group_first_size'} if defined
                            $params{$gname.'_group_first_size'};
                            $params{$tsn.'_size'} = $params{$gname.'_group_size'} if defined
                            $params{$gname.'_group_size'};
                            $params{$tsn.'_nolg'} = $params{$gname.'_nolg'} if defined
                            $params{$gname.'_nolg'};
                        }
                    }

                    $mostts =
                    "$params{'ts_temp_names'}, $params{'ts_data_names'}, $params{'ts_index_names'}";
                    $allts =
                    "ts_sys, ts_log, ts_undo, $params{'ts_temp_names'}, $params{'ts_data_names'}, $params{'
                    ts_index_names'}";
                    @ts_data = split(/,/, $params{'ts_data_names'});
                    @ts_index = split(/,/, $params{'ts_index_names'});
                    @ts_temp = split(/,/, $params{'ts_temp_names'});

```

```

ts_most = split(/,/, $mostts);
@ts_all = split(/,/, $allts);

foreach $ts_entry (@ts_most)
{
    $params{$ts_entry.'_#files'} = $params{'ts_def_#files'} if !defined
$params{$ts_entry.'_#files'};
    $params{$ts_entry.'_size'} = $params{'ts_def_size'} if !defined
$params{$ts_entry.'_size'};
    $params{$ts_entry.'_storage'} = $params{'ts_def_storage'} if !defined
$params{$ts_entry.'_storage'};
}

foreach $ts_entry (@ts_all)
{
    $params{$ts_entry.'_area'} = $params{'ts_def_area'} if !defined
$params{$ts_entry.'_area'};
    $params{$ts_entry.'_options'} = $params{'ts_def_options'} if !defined
$params{$ts_entry.'_options'};
    $params{$ts_entry.'_first_size'} = $params{$ts_entry.'_size'} if !defined
$params{$ts_entry.'_first_size'};
    $params{$ts_entry.'_temporary'} = "false" if !defined
$params{$ts_entry.'_temporary'};
    $params{$ts_entry.'_managed'} = "user" if !defined $params{$ts_entry.'_managed'};
    if (!defined $params{$ts_entry.'_datafiles'})
    {
        for ($i = 0; $i < $params{$ts_entry.'_#files'}; $i++)
        {
            ($i+1) == $params{$ts_entry.'_#files'} ? ($addcomma = "") :
($addcomma = ",");
            $nextfile = sprintf ("%s%d.f%s", $ts_entry, $i+1, $addcomma);
            $params{$ts_entry.'_datafiles'} = $params{$ts_entry.'_datafiles'}
. $nextfile;
        }
        @ts_datafiles = split(/,/, $params{$ts_entry.'_datafiles'});
        $cntr = 1;
        if ($ts_datafiles[0] =~ /\#/) # we want to replace all #'s
        {
            $filenm = shift(@ts_datafiles);
            $savename = $filenm;
            $params{$ts_entry.'_datafiles'} = "";
            for ($i = 0; $i < $params{$ts_entry.'_#files'}; $i++)
            {
                $filenm =~ s/\#/$cntr/g;
                $cntr++;
                ($i+1) == $params{$ts_entry.'_#files'} ? ($addcomma = "") :
($addcomma = ",");
                $params{$ts_entry.'_datafiles'} = $params{$ts_entry.'_datafiles'}
. $filenm . $addcomma;
                $filenm = $savename;
            }
            printf ("Expanded $savename to:\n%s\n%s\n", $params{$ts_entry.'_datafiles'},
$params{$ts_entry.'_datafiles'});
        }
        else
        {
            if (@ts_datafiles != $params{$ts_entry.'_#files'})
            {
                print "Number of files ($params{$ts_entry.'_#files'}) for
$ts_entry doesn't match\n_datafile parameter $params{$ts_entry.'_datafiles'}\n";
                exit(-1);
            }
        }
    }
    $params{'io_control_files'} = $params{'ts_sys_area'}.'t_cf1.f' if !defined
$params{'io_control_files'};

    @tab_list = split(/,/, $params{'tab_tables'});
    foreach $table (@tab_list)
    {
        $params{'tab_'.$table.'_ts'} = "ts_data" if !defined
$params{'tab_'.$table.'_ts'};
        $params{'tab_'.$table.'_load_degpar'} = $params{'load_deg_parallel'} if
!defined $params{'tab_'.$table.'_load_degpar'};
        $params{'tab_'.$table.'_#rows'} = $params{'scale_factor'};
    }

    @view_list = split(/,/, $params{'view_views'});
    foreach $view (@view_list)
    {
        $params{'view_'.$view.'_ts'} = "ts_data" if !defined
$params{'view_'.$view.'_ts'};
        $params{'view_'.$view.'_load_degpar'} = $params{'load_deg_parallel'} if
!defined $params{'view_'.$view.'_load_degpar'};
        $params{'view_'.$view.'_#rows'} = $params{'scale_factor'};
    }

    @tablelog_list = split(/,/, $params{'tablelog_list'});

    @clulist = split(/,/, $params{'clu_clusters'});
    foreach $cluster (@clulist)
    {
        $params{'clu_'.$cluster.'_ts'} = "ts_data" if !defined
$params{'clu_'.$cluster.'_ts'};
    }

    @constlist = split(/,/, $params{'con_constraints'});
    foreach $const (@constlist)
    {
        $params{'con_'.$const.'_ts'} = "ts_index" if (!defined
$params{'con_'.$const.'_ts'}) && ($params{'skip_ts'} != /index/);
    }

    @indexlist = split(/,/, $params{'ind_indices'});
    foreach $index (@indexlist)
    {
        $params{'ind_'.$index.'_ts'} = "ts_index" if (!defined
$params{'ind_'.$index.'_ts'}) && ($params{'skip_ts'} != /index/);
    }

    $params{'analyze_type'} = "via package dbms.stats" if
!defined($params{'analyze_type'});
    @analyzlist = split(/,/, $params{'anl_objects'});
    @analyzlist = "schema" if (@analyzlist == 0);
    foreach $object (@analyzlist)
    {
        $params{'anl_'.$object.'_object_type'} = "schema" if ($object =~ /schema/);
        $params{'anl_'.$object.'_object_type'} = "table" if !defined
$params{'anl_'.$object.'_object_type'};
        $params{'anl_'.$object.'_anl_type'} = "gather" if !defined
$params{'anl_'.$object.'_anl_type'};
    }

    @choblist = split(/,/, $params{'chob_objects'});
    foreach $object (@choblist)
    {
        $choblist{$object} = $params{'chob_'.$object.'_pardeg'};
    }

    @chstorlist = split(/,/, $params{'chstor_objects'});
    foreach $object (@chstorlist)
    {
        $chstorlist{$object} = $params{'chstor_'.$object.'_next'};
    }

    if ((defined $params{'verbose'}) && ($params{'verbose'} ==
/[tT][rR][uU][eE]/) || ($params{'verbose'} == 1)){
        $verbose=1;
    }
    else {
        $verbose=0;
    }
    if ((defined $params{'log_output'}) && ($params{'log_output'} ==
/[tT][rR][uU][eE]/) || ($params{'log_output'} == 1)){
        $log_output=1;
    }
    else {
        $log_output=0;
    }
    $params{'picre_recreate_extended_partitions'} = "false" if !defined
($params{'picre_recreate_extended_partitions'});
}

sub assigndefs
{
    while ($onedef = shift(@defparams))
    {
        $onedef =~ /^(.*)=(.*)$/;
        $key = $1;
        $value = $2;
        $params{$key} = $value if !defined $params{$key};
    }
}

sub auxdefaults
{
    @defparams = ();
    push (@defparams, 'scale_factor=0.1');
    # push (@defparams, 'dbs_area=?dbs/');
    push (@defparams, 'startupfile_dbcre=init.ora');
    push (@defparams, 'startupfile_shutd=none');
    push (@defparams, 'startupfile_start=none');
    push (@defparams, 'startupfile_dngen=init.ora');
    push (@defparams, 'startupfile_scocre=init.ora');
    push (@defparams, 'startupfile_scuto=init.ora');
    push (@defparams, 'startupfile_scuvo=init.ora');
    push (@defparams, 'startupfile_sctso=init.ora');
    push (@defparams, 'startupfile_dapop=init.ora');
    push (@defparams, 'startupfile_ixcre=init.ora');
    push (@defparams, 'startupfile_anlyz=init.ora');
    push (@defparams, 'startupfile_chob=init.ora');
    push (@defparams, 'startupfile_expln=init.ora');
    push (@defparams, 'startupfile_query=init.ora');
    push (@defparams, 'dbs_area=tmp');
    push (@defparams, 'dd_sql_area=?rdbms/admin/');
    push (@defparams, 'max_bgs=1');
    push (@defparams, 'user=bmuser');
    push (@defparams, 'passwd=mpasswd');
    push (@defparams, 'load_type=dellm');
    push (@defparams, 'load_field_terminator=whitespace');
    push (@defparams, 'ts_def_#files=1');
    push (@defparams, 'ts_def_options=reuse');
    push (@defparams, 'ts_def_first_size=1m');
    push (@defparams, 'ts_def_size=1m');
    push (@defparams, 'ts_sys_#files=1');
    push (@defparams, 'ts_log_#threads=1');
    push (@defparams, 'ts_log_#files_pt=2');
    push (@defparams, 'ts_undo_#files=1');
    push (@defparams, 'ts_sys_size=25m');
    push (@defparams, 'ts_log_size=2m');
    push (@defparams, 'ts_undo_size=10m');
    # push (@defparams, 'ts_data_names=ts_data');
    push (@defparams, 'load_insert_type=append');
    push (@defparams, 'load_deg_parallel=1');
    push (@defparams, 'special machine=');
    push (@defparams, 'shutdown_level=abort');
    push (@defparams, 'def_num_inst=1');
    # push (@defparams, 'privileges=resource,unlimited tablespace,connect');
    push (@defparams, 'privileges=DBA');
    push (@defparams, 'qry_number_trials=1');
}

sub tpcd
{
    @defparams = ();
    push (@defparams, 'user=tpcd');
    push (@defparams, 'passwd=tpcd');
    'tab_tables=lineitem,orders,partsupp,parts,customer,supplier,nation,region';
    push (@defparams, 'view_views=mav_11_mv_locs');
    push (@defparams, 'viewlog_list=lineitem,orders');
    # push (@defparams, 'may_11_columns=count p group,1 shipdate,1 l_linestatus,sum qty,sum bas
e price,sum disc price,count disc price,sum charge,
count charge,count qty,count p_price,sum disc,count_disc,count_order');
    push (@defparams, 'tab_lineitem_columns=1 orderkey,1 partkey,1 suppley,1 linenumber,1 quantity,1 ext
endedprice,1 discount,1 tax,1 returnflag,1 l_linestatus,1 shipdate,1 committdate,1 re
ceipdate,1 shipinstruct,1 shipmode,1 comment');
    # push (@defparams, 'tab_orders_columns=1 orderkey,o_custkey,o_orderstatus,o_totalprice,o_orderdate,o
orderpriority,o_clerk,o_shippriority,o_comment');
    push (@defparams, 'tab_partsupp_columns=ps_partkey,ps_suppley,ps_availqty,ps_supplycost,ps_comment');
    # push (@defparams, 'tab_parts_columns=sp_partkey,p_name,p_mfgp,p_brand,p_type,p_size,p_container,p_ret
ailprice,p_comment');
    push (@defparams, 'tab_customer_columns=c_custkey,c_name,c_address,c_nationkey,c_phone,c_acctbal,c_mv
ktsegment,c_comment');
}

```

```

push (@defparams, 'tab_supplier_columns=s_supplier,s_name,s_address,s_nationkey,s_phone,s_acctbal,s_c
comment');
push (@defparams, 'tab_nation_columns=n_nationkey,n_name,n_regionkey,n_comment');
push (@defparams, 'tab_region_columns=r_regionkey,r_name,r_comment');
push (@defparams, 'tab_lineitem_rows=6100000'); # Somewhat of a random value
push (@defparams, 'tab_orders_rows=1500000');
push (@defparams, 'tab_partsupp_rows=800000');
push (@defparams, 'tab_parts_rows=200000');
push (@defparams, 'tab_customer_rows=150000');
push (@defparams, 'tab_supplier_rows=10000');
push (@defparams, 'tab_nation_rows=0');
push (@defparams, 'tab_region_rows=0');
push (@defparams, 'tab_lineitem_parttype=range');
push (@defparams, 'tab_lineitem_l_orderkey_type=number');
push (@defparams, 'tab_lineitem_l_partkey_type=number');
push (@defparams, 'tab_lineitem_l_suppkey_type=number');
push (@defparams, 'tab_lineitem_l_linenumber_type=number');
push (@defparams, 'tab_lineitem_l_quantity_type=number');
push (@defparams, 'tab_lineitem_l_extendedprice_type=number');
push (@defparams, 'tab_lineitem_l_discount_type=number');
push (@defparams, 'tab_lineitem_l_tax_type=number');
push (@defparams, 'tab_lineitem_l_returnflag_type=char(1)');
push (@defparams, 'tab_lineitem_l_linestatus_type=char(1)');
push (@defparams, 'tab_lineitem_l_shipdate_type=date');
push (@defparams, 'tab_lineitem_l_commitdate_type=date');
push (@defparams, 'tab_lineitem_l_receiptdate_type=date');
push (@defparams, 'tab_lineitem_l_shipinstruct_type=varchar(25)');
push (@defparams, 'tab_lineitem_l_shipmode_type=varchar(10)');
push (@defparams, 'tab_lineitem_l_comment_type=varchar(44)');
push (@defparams, 'tab_lineitem_l_orderkey_pos=13:24');
push (@defparams, 'tab_lineitem_l_partkey_pos=25:36');
push (@defparams, 'tab_lineitem_l_suppkey_pos=37:48');
push (@defparams, 'tab_lineitem_l_linenumber_pos=49:60');
push (@defparams, 'tab_lineitem_l_quantity_pos=61:72');
push (@defparams, 'tab_lineitem_l_extendedprice_pos=73:87');
push (@defparams, 'tab_lineitem_l_discount_pos=88:102');
push (@defparams, 'tab_lineitem_l_tax_pos=103:117');
push (@defparams, 'tab_lineitem_l_returnflag_pos=118:118');
push (@defparams, 'tab_lineitem_l_linestatus_pos=120:120');
push (@defparams, 'tab_lineitem_l_shipdate_pos=122:131');
push (@defparams, 'tab_lineitem_l_commitdate_pos=135:144');
push (@defparams, 'tab_lineitem_l_receiptdate_pos=148:157');
push (@defparams, 'tab_lineitem_l_shipinstruct_pos=161:185');
push (@defparams, 'tab_lineitem_l_shipmode_pos=186:195');
push (@defparams, 'tab_lineitem_l_comment_pos=196:239');
push (@defparams, 'tab_lineitem_l_shipdate_loadcolx=date "yyyy-mm-dd"');
push (@defparams, 'tab_lineitem_l_commitdate_loadcolx=date "yyyy-mm-dd"');
push (@defparams, 'tab_lineitem_l_receiptdate_loadcolx=date "yyyy-mm-dd"');
push (@defparams, 'tab_orders_parttype=range');
push (@defparams, 'tab_orders_o_orderkey_type=number');
push (@defparams, 'tab_orders_o_custkey_type=number');
push (@defparams, 'tab_orders_o_orderstatus_type=char(1)');
push (@defparams, 'tab_orders_o_totalprice_type=number');
push (@defparams, 'tab_orders_o_orderdate_type=date');
push (@defparams, 'tab_orders_o_orderpriority_type=varchar(15)');
push (@defparams, 'tab_orders_o_clerk_type=varchar(15)');
push (@defparams, 'tab_orders_o_shippriority_type=number');
push (@defparams, 'tab_orders_o_comment_type=varchar(79)');
push (@defparams, 'tab_orders_o_orderkey_pos=13:24');
push (@defparams, 'tab_orders_o_custkey_pos=25:36');
push (@defparams, 'tab_orders_o_orderstatus_pos=37:37');
push (@defparams, 'tab_orders_o_totalprice_pos=39:53');
push (@defparams, 'tab_orders_o_orderdate_pos=54:63');
push (@defparams, 'tab_orders_o_orderpriority_pos=67:81');
push (@defparams, 'tab_orders_o_clerk_pos=82:96');
push (@defparams, 'tab_orders_o_shippriority_pos=97:108');
push (@defparams, 'tab_orders_o_comment_pos=109:187');
push (@defparams, 'tab_orders_o_orderdate_loadcolx=date "yyyy-mm-dd"');
push (@defparams, 'tab_partsupp_parttype=range');
push (@defparams, 'tab_partsupp_ps_partkey_type=number');
push (@defparams, 'tab_partsupp_ps_suppkey_type=number');
push (@defparams, 'tab_partsupp_ps_availqty_type=number');
push (@defparams, 'tab_partsupp_ps_supplycost_type=number');
push (@defparams, 'tab_partsupp_ps_comment_type=varchar(199)');
push (@defparams, 'tab_partsupp_ps_partkey_pos=1:12');
push (@defparams, 'tab_partsupp_ps_suppkey_pos=13:24');
push (@defparams, 'tab_partsupp_ps_availqty_pos=25:36');
push (@defparams, 'tab_partsupp_ps_supplycost_pos=37:51');
push (@defparams, 'tab_partsupp_ps_comment_pos=52:250');
push (@defparams, 'tab_parts_parttype=range');
push (@defparams, 'tab_parts_p_partkey_type=number');
push (@defparams, 'tab_parts_p_name_type=varchar(55)');
push (@defparams, 'tab_parts_p_mfgr_type=varchar(25)');
push (@defparams, 'tab_parts_p_brand_type=varchar(10)');
push (@defparams, 'tab_parts_p_type_type=varchar(25)');
push (@defparams, 'tab_parts_p_size_type=number');
push (@defparams, 'tab_parts_p_container_type=varchar(10)');
push (@defparams, 'tab_parts_p_retailprice_type=number');
push (@defparams, 'tab_parts_p_comment_pos=165:187');
push (@defparams, 'tab_customer_parttype=range');
push (@defparams, 'tab_customer_c_custkey_type=number');
push (@defparams, 'tab_customer_c_name_type=varchar(25)');
push (@defparams, 'tab_customer_c_address_type=varchar(40)');
push (@defparams, 'tab_customer_c_nationkey_type=number');
push (@defparams, 'tab_customer_c_phone_type=varchar(15)');
push (@defparams, 'tab_customer_c_acctbal_type=number');
push (@defparams, 'tab_customer_c_mktsegment_type=varchar(10)');
push (@defparams, 'tab_customer_c_comment_type=varchar(117)');
push (@defparams, 'tab_customer_c_custkey_pos=1:12');
push (@defparams, 'tab_customer_c_name_pos=13:30');
push (@defparams, 'tab_customer_c_address_pos=31:70');

push (@defparams, 'tab_customer_c_nationkey_pos=71:82');
push (@defparams, 'tab_customer_c_phone_pos=83:97');
push (@defparams, 'tab_customer_c_acctbal_pos=98:112');
push (@defparams, 'tab_customer_c_mktsegment_pos=113:122');
push (@defparams, 'tab_customer_c_comment_pos=123:239');
push (@defparams, 'tab_supplier_parttype=range');
push (@defparams, 'tab_supplier_s_suppkey_type=number');
push (@defparams, 'tab_supplier_s_name_type=varchar(25)');
push (@defparams, 'tab_supplier_s_address_type=varchar(40)');
push (@defparams, 'tab_supplier_s_nationkey_type=number');
push (@defparams, 'tab_supplier_s_phone_type=varchar(15)');
push (@defparams, 'tab_supplier_s_acctbal_type=number');
push (@defparams, 'tab_supplier_s_comment_type=varchar(101)');
push (@defparams, 'tab_supplier_s_suppkey_pos=1:12');
push (@defparams, 'tab_supplier_s_name_pos=13:37');
push (@defparams, 'tab_supplier_s_address_pos=38:77');
push (@defparams, 'tab_supplier_s_nationkey_pos=78:89');
push (@defparams, 'tab_supplier_s_phone_pos=90:104');
push (@defparams, 'tab_supplier_s_acctbal_pos=105:119');
push (@defparams, 'tab_supplier_s_comment_pos=120:220');
push (@defparams, 'tab_nation_parttype=range');
push (@defparams, 'tab_nation_n_nationkey_type=number');
push (@defparams, 'tab_nation_n_name_type=varchar(25)');
push (@defparams, 'tab_nation_n_regionkey_type=number');
push (@defparams, 'tab_nation_n_comment_type=varchar(152)');
push (@defparams, 'tab_nation_n_nationkey_pos=1:12');
push (@defparams, 'tab_nation_n_name_pos=13:37');
push (@defparams, 'tab_nation_n_regionkey_pos=38:49');
push (@defparams, 'tab_nation_n_comment_pos=50:164');
push (@defparams, 'tab_region_r_regionkey_type=number');
push (@defparams, 'tab_region_r_name_type=varchar(25)');
push (@defparams, 'tab_region_r_comment_type=varchar(152)');
push (@defparams, 'tab_region_r_regionkey_pos=1:12');
push (@defparams, 'tab_region_r_name_pos=13:37');
push (@defparams, 'tab_region_r_comment_pos=38:152');

push (@defparams, 'load_dbgen_output_prefix='. $params{'load_flatfile_area'}.'lineitem');
push (@defparams, 'load_dbgen_orders_output_prefix='. $params{'load_flatfile_area'}.'orders');
push (@defparams, 'load_dbgen_customer_output_prefix='. $params{'load_flatfile_area'}.'customer');
push (@defparams, 'load_dbgen_parts_output_prefix='. $params{'load_flatfile_area'}.'parts');
push (@defparams, 'load_dbgen_partsupp_output_prefix='. $params{'load_flatfile_area'}.'partsupp');
push (@defparams, 'load_dbgen_supplier_output_prefix='. $params{'load_flatfile_area'}.'supplier');
push (@defparams, 'load_dbgen_nation_output_prefix='. $params{'load_flatfile_area'}.'nation');
push (@defparams, 'load_dbgen_region_output_prefix='. $params{'load_flatfile_area'}.'region');
push (@defparams, 'load_type=fixed');
}

sub wisc
{
    @defparams = ();
    push (@defparams, 'user=wisc');
    push (@defparams, 'passwd=wisc');
    push (@defparams, 'tab_tables=wisc');
    push (@defparams, 'load_field_terminator=whitespace');
    push (@defparams, 'tab_wisc_columns=unique1,unique2,two,four,tен,ten,twenty,hundred,thousand,fivethous,t
enthous,oddi00,even00,stringul,stringu2,stringu4');
    push (@defparams, 'tab_wisc_rows=1000000');
    push (@defparams, 'tab_wisc_unique1_type=number');
    push (@defparams, 'tab_wisc_unique2_type=number');
    push (@defparams, 'tab_wisc_two_type=number');
    push (@defparams, 'tab_wisc_four_type=number');
    push (@defparams, 'tab_wisc_ten_type=number');
    push (@defparams, 'tab_wisc_twenty_type=number');
    push (@defparams, 'tab_wisc_hundred_type=number');
    push (@defparams, 'tab_wisc_thousand_type=number');
    push (@defparams, 'tab_wisc_fivethous_type=number');
    push (@defparams, 'tab_wisc_tenthous_type=number');
    push (@defparams, 'tab_wisc_oddi00_type=number');
    push (@defparams, 'tab_wisc_even00_type=number');
    push (@defparams, 'tab_wisc_stringul_type=varchar(52)');
    push (@defparams, 'tab_wisc_stringu2_type=varchar(52)');
    push (@defparams, 'tab_wisc_stringu4_type=varchar(52)');
    push (@defparams, 'tab_wisc_unique1_loadcolx=integer external');
    push (@defparams, 'tab_wisc_unique2_loadcolx=integer external');
    push (@defparams, 'tab_wisc_two_loadcolx=integer external');
    push (@defparams, 'tab_wisc_four_loadcolx=integer external');
    push (@defparams, 'tab_wisc_ten_loadcolx=integer external');
    push (@defparams, 'tab_wisc_twenty_loadcolx=integer external');
    push (@defparams, 'tab_wisc_hundred_loadcolx=integer external');
    push (@defparams, 'tab_wisc_thousand_loadcolx=integer external');
    push (@defparams, 'tab_wisc_fivethous_loadcolx=integer external');
    push (@defparams, 'tab_wisc_tenthous_loadcolx=integer external');
    push (@defparams, 'tab_wisc_oddi00_loadcolx=integer external');
    push (@defparams, 'tab_wisc_even00_loadcolx=integer external');
    push (@defparams, 'tab_wisc_stringul_loadcolx=char(52)');
    push (@defparams, 'tab_wisc_stringu2_loadcolx=char(52)');
    push (@defparams, 'tab_wisc_stringu4_loadcolx=char(52)');
}

sub time0
{
    if (($os cmp "nt") == 0)
    {
        $value = shift (@_);
        $value = "%time%" . $value;
        $value = $value . "\n" if ($value != /.*\n$/);
        print OUTFILE "$value";
        $value = "";
    }
}

sub read_parameter_description
{
    @paramdesc = ();
    stat ("%$pid%");
    if (-e _)

```

```

{
open (PDFFILE, "$pdfile");
while (<PDFILE>)
{
($key,$desc)=split (/:/, $_, 2);
# $key = shift(@1);
$key =~ s/(\w*)<(\w*)>(\w*)/$1\\w*$3/g;
$paramdesc{$key} = $desc;
}
}
else
{
print "Parameterfile: $pdfile does not exist\n will continue without ... \n";
}
}

sub key_exists
{
$result = 1;
return if ($rsilent == 1);
@keys = keys %paramdesc;
$х = $_[0]."!";
while ($#keys >=0)
{
$key = pop(@keys);
$key=$key."!";
return if ($х =~ /$key/);
}
$result = -1;
}

```

tera_anlyz.dat

```

#####
# preprocessing-like directives

%b-preproc

*sql
\echo "{}" > script*getenv(BUMPX_CTR).sql
\svrmgrl <<|
\set echo on;
\set termout on;
\connect internal;
\select to_char(sysdate, 'MM-DD-YYYY HH24:MI:SS') now from dual;
\@script*getenv(BUMPX_CTR).sql;
\select to_char(sysdate, 'MM-DD-YYYY HH24:MI:SS') now from dual;
\exit;
\!
\bin/rm script*getenv(BUMPX_CTR).sql;

*load
\sqlldr {}

*mknod
\mknod {}

*dbgen
\setenv DSS_PATH {}
\ohome/100g/tpcd/dbgen/dbgen {}

*check_stop
\ohome/100g/checkstop.sh

*time
\echo
\echo {}
\date
\echo

*sh
\{}

%e-preproc
%b-anlyz
%bgon=15
#####
# Analyze Phase
*time
{
START ANALYZING SCHEMA
}
*sql
{
connect tpcd/tpcd;
execute dbms_stats.gather_schema_stats('tpcd' , estimate_percent => 1 , degree =>
64 , granularity => 'GLOBAL' , block_sample => 'true');
}
*wait
*time
*sql
{
execute dbms_stats.gather_schema_stats('sys' , estimate_percent => 0.00001);
}
*wait
*time
*sql
{
connect tpcd/tpcd;
execute dbms_stats.delete_table_stats('tpcd' , 'MLOG$_lineitem');
}
*sql
{

```

```

connect tpcd/tpcd;
execute dbms_stats.delete_table_stats('tpcd' , 'MLOG$_orders');
}
*sql
{
connect tpcd/tpcd;
execute dbms_stats.delete_table_stats('tpcd' , 'MLOG$_customer');
}
*sql
{
connect tpcd/tpcd;
execute dbms_stats.delete_table_stats('tpcd' , 'MLOG$_supplier');
}
*sql
{
connect tpcd/tpcd;
execute dbms_stats.delete_table_stats('tpcd' , 'MLOG$_parts');
}
*sql
{
connect tpcd/tpcd;
execute dbms_stats.delete_table_stats('tpcd' , 'MLOG$_partsupp');
}
*sql
{
connect tpcd/tpcd;
execute dbms_stats.delete_table_stats('tpcd' , 'MLOG$_nation');
}
*sql
{
connect tpcd/tpcd;
execute dbms_stats.delete_table_stats('tpcd' , 'MLOG$_region');
}
*sql
{
connect tpcd/tpcd;
execute dbms_stats.delete_table_stats('tpcd' , 'MLOG$_mj_plocs');
}
*wait
*sql
{
connect tpcd/tpcd;
@/ora81/oracle814/980901/rdbms/admin/utlxplan.sql;
}
*wait
*time
{
DONE ANALYZING SCHEMA
}
*bgoff
%e-anlyz

```

tera_chob.dat

```

#####
# preprocessing-like directives

%b-preproc

*sql
\echo "{}" > script*getenv(BUMPX_CTR).sql
\svrmgrl <<|
\set echo on;
\set termout on;
\connect internal;
\select to_char(sysdate, 'MM-DD-YYYY HH24:MI:SS') now from dual;
\@script*getenv(BUMPX_CTR).sql;
\select to_char(sysdate, 'MM-DD-YYYY HH24:MI:SS') now from dual;
\exit;
\!
\bin/rm script*getenv(BUMPX_CTR).sql;

*load
\sqlldr {}

*mknod
\mknod {}

*dbgen
\setenv DSS_PATH {}
\ohome/100g/tpcd/dbgen/dbgen {}

*check_stop
\ohome/100g/checkstop.sh

*time
\echo
\echo {}
\date
\echo

*sh
\{}

%e-preproc
%b-chob
%bgon=160
#####b#####
# Make CHOB Phase
*sql
{
connect tpcd/tpcd;
alter table lineitem parallel 160;
alter table lineitem storage (next 10m);
alter table orders parallel 160;
alter table orders storage (next 10m);
alter table mj_ppssrn parallel 160;
alter table mj_plocs parallel 160;

```

```

alter table mj_plocs storage (next 20m);
alter index ij_l_lrid parallel 160;
alter index ij_l_lrid storage (next 10m);
alter index ij_l_odate parallel 160;
alter index ij_l_odate storage (next 20m);
alter index ij_l_orid parallel 160;
alter index ij_l_orid storage (next 10m);
alter index ij_l_pkey parallel 160;
alter index ij_l_pkey storage (next 20m);
alter index ij_l_ptp storage (next 20m);
alter index ij_l_ptp parallel 40;
alter index ij_l_rdate parallel 160;
alter index ij_l_rdate storage (next 20m);
alter index ij_l_sdate parallel 160;
alter index ij_l_sdate storage (next 10m);
alter index l_ored parallel 160;
alter index l_ored storage (next 10m);
alter index o_clokod parallel 16;
alter index o_clokod storage (next 10m);
alter index o_okey parallel 160;
alter index o_okey storage (next 10m);
alter index ij_l_nullrid parallel 160;
alter index ij_l_nullrid storage (next 1m);
alter table MLOG$\_lineitem parallel 160;
alter table MLOG$\_orders parallel 160;
alter table MLOG$\_mj_plocs parallel 160;
}
*wait
*time
{
DONE BUILDING DATABASE
}
*sql
{
shutdown;
}
*wait
*bgoff
%e-chob

```

tera_dapop_noord.dat

```

#####
# preprocessing-like directives

%b-preproc

*sql
\echo "{}" > script*getenv(BUMPX_CTR).sql
\svrmgrl <<|
\set echo on;
\set termout on;
\connect internal;
\select to_char(sysdate, 'MM-DD-YYYY HH24:MI:SS') now from dual;
\script*getenv(BUMPX_CTR).sql;
\select to_char(sysdate, 'MM-DD-YYYY HH24:MI:SS') now from dual;
\exit;
\!
\bin/rm script*getenv(BUMPX_CTR).sql;

*load
\sqlldr {}

*mknod
\mknod {}

*dbgen
\setenv DSS_PATH {}
\ohome/100g/tpcd/dbgen/dbgen {}

*check_stop
\ohome/100g/checkstop.sh

*time
\echo
\echo {}
\date
\echo

*sh
\{}

%e-preproc
%b-dapop
*bgcn=80
#####
# Database Population Phase
*time
{
START LOADING LINEITEM
}
*wait
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/lineitem1.ctl
lbg=/ora81/scripts/load/log/lineitem1.log
bad=/ora81/scripts/load/log/lineitem1.bad direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/lineitem2.ctl
lbg=/ora81/scripts/load/log/lineitem2.log
bad=/ora81/scripts/load/log/lineitem2.bad direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/lineitem3.ctl
lbg=/ora81/scripts/load/log/lineitem3.log
bad=/ora81/scripts/load/log/lineitem3.bad direct=true parallel=true
}
}

```

```

*load
{
tpcd/tpcd control=/ora81/scripts/load/control/lineitem4.ctl
lbg=/ora81/scripts/load/log/lineitem4.log
bad=/ora81/scripts/load/log/lineitem4.bad direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/lineitem5.ctl
lbg=/ora81/scripts/load/log/lineitem5.log
bad=/ora81/scripts/load/log/lineitem5.bad direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/lineitem6.ctl
lbg=/ora81/scripts/load/log/lineitem6.log
bad=/ora81/scripts/load/log/lineitem6.bad direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/lineitem7.ctl
lbg=/ora81/scripts/load/log/lineitem7.log
bad=/ora81/scripts/load/log/lineitem7.bad direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/lineitem8.ctl
lbg=/ora81/scripts/load/log/lineitem8.log
bad=/ora81/scripts/load/log/lineitem8.bad direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/lineitem9.ctl
lbg=/ora81/scripts/load/log/lineitem9.log
bad=/ora81/scripts/load/log/lineitem9.bad direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/lineitem10.ctl
lbg=/ora81/scripts/load/log/lineitem10.log
bad=/ora81/scripts/load/log/lineitem10.bad direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/lineitem11.ctl
lbg=/ora81/scripts/load/log/lineitem11.log
bad=/ora81/scripts/load/log/lineitem11.bad direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/lineitem12.ctl
lbg=/ora81/scripts/load/log/lineitem12.log
bad=/ora81/scripts/load/log/lineitem12.bad direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/lineitem13.ctl
lbg=/ora81/scripts/load/log/lineitem13.log
bad=/ora81/scripts/load/log/lineitem13.bad direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/lineitem14.ctl
lbg=/ora81/scripts/load/log/lineitem14.log
bad=/ora81/scripts/load/log/lineitem14.bad direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/lineitem15.ctl
lbg=/ora81/scripts/load/log/lineitem15.log
bad=/ora81/scripts/load/log/lineitem15.bad direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/lineitem16.ctl
lbg=/ora81/scripts/load/log/lineitem16.log
bad=/ora81/scripts/load/log/lineitem16.bad direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/lineitem17.ctl
lbg=/ora81/scripts/load/log/lineitem17.log
bad=/ora81/scripts/load/log/lineitem17.bad direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/lineitem18.ctl
lbg=/ora81/scripts/load/log/lineitem18.log
bad=/ora81/scripts/load/log/lineitem18.bad direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/lineitem19.ctl
lbg=/ora81/scripts/load/log/lineitem19.log
bad=/ora81/scripts/load/log/lineitem19.bad direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/lineitem20.ctl
lbg=/ora81/scripts/load/log/lineitem20.log
bad=/ora81/scripts/load/log/lineitem20.bad direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/lineitem21.ctl
lbg=/ora81/scripts/load/log/lineitem21.log
bad=/ora81/scripts/load/log/lineitem21.bad direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/lineitem22.ctl
lbg=/ora81/scripts/load/log/lineitem22.log
bad=/ora81/scripts/load/log/lineitem22.bad direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/lineitem23.ctl
lbg=/ora81/scripts/load/log/lineitem23.log
bad=/ora81/scripts/load/log/lineitem23.bad direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/lineitem24.ctl
lbg=/ora81/scripts/load/log/lineitem24.log
bad=/ora81/scripts/load/log/lineitem24.bad direct=true parallel=true
}
}

```


[illegible]

[illegible]


```

}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/supplier.ctl
log=/ora81/scripts/load/log/supplier74.log
bad=/ora81/scripts/load/log/supplier.bad data=/fs/fs83/supplier.tbl.74 direct=true
parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/supplier.ctl
log=/ora81/scripts/load/log/supplier75.log
bad=/ora81/scripts/load/log/supplier.bad data=/fs/fs84/supplier.tbl.75 direct=true
parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/supplier.ctl
log=/ora81/scripts/load/log/supplier76.log
bad=/ora81/scripts/load/log/supplier.bad data=/fs/fs85/supplier.tbl.76 direct=true
parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/supplier.ctl
log=/ora81/scripts/load/log/supplier77.log
bad=/ora81/scripts/load/log/supplier.bad data=/fs/fs86/supplier.tbl.77 direct=true
parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/supplier.ctl
log=/ora81/scripts/load/log/supplier78.log
bad=/ora81/scripts/load/log/supplier.bad data=/fs/fs87/supplier.tbl.78 direct=true
parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/supplier.ctl
log=/ora81/scripts/load/log/supplier79.log
bad=/ora81/scripts/load/log/supplier.bad data=/fs/fs88/supplier.tbl.79 direct=true
parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/supplier.ctl
log=/ora81/scripts/load/log/supplier80.log
bad=/ora81/scripts/load/log/supplier.bad data=/fs/fs89/supplier.tbl.80 direct=true
parallel=true
}
}
*wait
*time
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/nation.ctl
log=/ora81/scripts/load/log/nation.log bad=/ora81/scripts/load/log/nation.bad
data=/ora81/scripts/load/flatfiles/nation.tbl direct=true
}
*wait
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/region.ctl
log=/ora81/scripts/load/log/region.log bad=/ora81/scripts/load/log/region.bad
data=/ora81/scripts/load/flatfiles/region.tbl direct=true
}
}
*wait
*time
{
DONE LOADING SMALL TABLES
}
}
*bgoff
%e-dapop

```

tera_dapop_orders.dat

```

#####
# preprocessing-like directives

%b-preproc

*sql
\echo "{}" > script*getenv (BUMPX_CTR).sql
\svrmgrl <<!
\set echo on;
\set termout on;
\connect internal;
\select to_char(sysdate, 'MM-DD-YYYY HH24:MI:SS') now from dual;
\@script*getenv (BUMPX_CTR).sql;
\select to_char(sysdate, 'MM-DD-YYYY HH24:MI:SS') now from dual;
\exit;
\!
\bin/rm script*getenv (BUMPX_CTR).sql;

*load
\sqlldr {}

*mknod
\mknod {}

*dbgen
\setenv DSS_PATH {}
\ohome/100g/tpcd/dbgen/dbgen {}

*check_stop
\ohome/100g/checkstop.sh

*time
\echo
\echo {}
\date
\echo

*sh
\{}

%e-preproc
%b-dapop

```

```

*bgon=24
#####
# Database Population Phase
*time
{
START LOADING ORDERS
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/orders1.ctl
log=/ora81/scripts/load/log/orders1.log bad=/ora81/scripts/load/log/orders1.bad
direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/orders2.ctl
log=/ora81/scripts/load/log/orders2.log bad=/ora81/scripts/load/log/orders2.bad
direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/orders3.ctl
log=/ora81/scripts/load/log/orders3.log bad=/ora81/scripts/load/log/orders3.bad
direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/orders4.ctl
log=/ora81/scripts/load/log/orders4.log bad=/ora81/scripts/load/log/orders4.bad
direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/orders5.ctl
log=/ora81/scripts/load/log/orders5.log bad=/ora81/scripts/load/log/orders5.bad
direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/orders6.ctl
log=/ora81/scripts/load/log/orders6.log bad=/ora81/scripts/load/log/orders6.bad
direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/orders7.ctl
log=/ora81/scripts/load/log/orders7.log bad=/ora81/scripts/load/log/orders7.bad
direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/orders8.ctl
log=/ora81/scripts/load/log/orders8.log bad=/ora81/scripts/load/log/orders8.bad
direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/orders9.ctl
log=/ora81/scripts/load/log/orders9.log bad=/ora81/scripts/load/log/orders9.bad
direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/orders10.ctl
log=/ora81/scripts/load/log/orders10.log bad=/ora81/scripts/load/log/orders10.bad
direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/orders11.ctl
log=/ora81/scripts/load/log/orders11.log bad=/ora81/scripts/load/log/orders11.bad
direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/orders12.ctl
log=/ora81/scripts/load/log/orders12.log bad=/ora81/scripts/load/log/orders12.bad
direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/orders13.ctl
log=/ora81/scripts/load/log/orders13.log bad=/ora81/scripts/load/log/orders13.bad
direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/orders14.ctl
log=/ora81/scripts/load/log/orders14.log bad=/ora81/scripts/load/log/orders14.bad
direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/orders15.ctl
log=/ora81/scripts/load/log/orders15.log bad=/ora81/scripts/load/log/orders15.bad
direct=true parallel=true
}
*load
{
tpcd/tpcd control=/ora81/scripts/load/control/orders16.ctl
log=/ora81/scripts/load/log/orders16.log bad=/ora81/scripts/load/log/orders16.bad
direct=true parallel=true
}
}
*wait
*time
{
END LOADING ORDERS
}
}
*wait
*bgoff
%e-dapop

```

tera_dbcre.dat

```

#####
# preprocessing-like directives

%b-preproc

```


[illegible][illegible]


```
*wait
*time
{
DONE PHASE 1 OF DB CREATION
}
*wait
*bgoff
%e-dbcre
```

tera_ixcre.dat

```
#####
# preprocessing-like directives

%b-preproc

*sql
\echo "{}" > script*getenv(BUMPX_CTR).sql
\svrmgrl <<|
\set echo on;
\set termout on;
\connect internal;
\select to_char(sysdate, 'MM-DD-YYYY HH24:MI:SS') now from dual;
\@script*getenv(BUMPX_CTR).sql;
\select to_char(sysdate, 'MM-DD-YYYY HH24:MI:SS') now from dual;
\exit;
\!
\bin/rm script*getenv(BUMPX_CTR).sql;

*load
\sqlldr {}

*mknod
\mknod {}

*dbgen
\setenv DSS_PATH {}
\ohome/100g/tpcd/dbgen/dbgen {}

*check_stop
\ohome/100g/checkstop.sh

*time
\echo
\echo {}
\date
\echo

*sh
\{}

%e-preproc
%b-ixcre
*bgon-1
#####
# Index Creation Phase
*time
{
START INDEX CREATION
}
*sql
{
shutdown;
}
*wait
*sql
{
startup pfile=ora81/oracle814/980901/dbs/index_build.ora;
}
*wait
*time
*sql
{
connect tpcd/tpcd;
drop index l_ored;
create index l_ored
on lineitem (l_orderkey,l_returnflag,l_extendedprice,l_discount)
pctfree 2
initrans 10
nologging
compute statistics
tablespace ts_ind_lored
storage (initial 100m next 100m freelists 32 freelist groups 5 maxextents
unlimited pctincrease 0)
parallel (degree 160 )
global partition by range (l_orderkey)
(
partition lored1 values less than (37500001)
,
partition lored2 values less than (75000001)
,
partition lored3 values less than (112500001)
,
partition lored4 values less than (150000001)
,
partition lored5 values less than (187500001)
,
partition lored6 values less than (225000001)
,
partition lored7 values less than (262500001)
,
partition lored8 values less than (300000001)
,
partition lored9 values less than (337500001)
,
partition lored10 values less than (375000001)
,
partition lored11 values less than (412500001)
,
```

```
partition lored12 values less than (450000001)
,
partition lored13 values less than (487500001)
,
partition lored14 values less than (525000001)
,
partition lored15 values less than (562500001)
,
partition lored16 values less than (600000001)
,
partition lored17 values less than (637500001)
,
partition lored18 values less than (675000001)
,
partition lored19 values less than (712500001)
,
partition lored20 values less than (750000001)
,
partition lored21 values less than (787500001)
,
partition lored22 values less than (825000001)
,
partition lored23 values less than (862500001)
,
partition lored24 values less than (900000001)
,
partition lored25 values less than (937500001)
,
partition lored26 values less than (975000001)
,
partition lored27 values less than (1012500001)
,
partition lored28 values less than (1050000001)
,
partition lored29 values less than (1087500001)
,
partition lored30 values less than (1125000001)
,
partition lored31 values less than (1162500001)
,
partition lored32 values less than (1200000001)
,
partition lored33 values less than (1237500001)
,
partition lored34 values less than (1275000001)
,
partition lored35 values less than (1312500001)
,
partition lored36 values less than (1350000001)
,
partition lored37 values less than (1387500001)
,
partition lored38 values less than (1425000001)
,
partition lored39 values less than (1462500001)
,
partition lored40 values less than (1500000001)
,
partition lored41 values less than (1537500001)
,
partition lored42 values less than (1575000001)
,
partition lored43 values less than (1612500001)
,
partition lored44 values less than (1650000001)
,
partition lored45 values less than (1687500001)
,
partition lored46 values less than (1725000001)
,
partition lored47 values less than (1762500001)
,
partition lored48 values less than (1800000001)
,
partition lored49 values less than (1837500001)
,
partition lored50 values less than (1875000001)
,
partition lored51 values less than (1912500001)
,
partition lored52 values less than (1950000001)
,
partition lored53 values less than (1987500001)
,
partition lored54 values less than (2025000001)
,
partition lored55 values less than (2062500001)
,
partition lored56 values less than (2100000001)
,
partition lored57 values less than (2137500001)
,
partition lored58 values less than (2175000001)
,
partition lored59 values less than (2212500001)
,
partition lored60 values less than (2250000001)
,
partition lored61 values less than (2287500001)
,
partition lored62 values less than (2325000001)
,
partition lored63 values less than (2362500001)
,
partition lored64 values less than (2400000001)
,
partition lored65 values less than (2437500001)
,
partition lored66 values less than (2475000001)
,
```

```

partition lored67 values less than (2512500001)
,
partition lored68 values less than (2550000001)
,
partition lored69 values less than (2587500001)
,
partition lored70 values less than (2625000001)
,
partition lored71 values less than (2662500001)
,
partition lored72 values less than (2700000001)
,
partition lored73 values less than (2737500001)
,
partition lored74 values less than (2775000001)
,
partition lored75 values less than (2812500001)
,
partition lored76 values less than (2850000001)
,
partition lored77 values less than (2887500001)
,
partition lored78 values less than (2925000001)
,
partition lored79 values less than (2962500001)
,
partition lored80 values less than (3000000001)
,
partition lored81 values less than (3037500001)
,
partition lored82 values less than (3075000001)
,
partition lored83 values less than (3112500001)
,
partition lored84 values less than (3150000001)
,
partition lored85 values less than (3187500001)
,
partition lored86 values less than (3225000001)
,
partition lored87 values less than (3262500001)
,
partition lored88 values less than (3300000001)
,
partition lored89 values less than (3337500001)
,
partition lored90 values less than (3375000001)
,
partition lored91 values less than (3412500001)
,
partition lored92 values less than (3450000001)
,
partition lored93 values less than (3487500001)
,
partition lored94 values less than (3525000001)
,
partition lored95 values less than (3562500001)
,
partition lored96 values less than (3600000001)
,
partition lored97 values less than (3637500001)
,
partition lored98 values less than (3675000001)
,
partition lored99 values less than (3712500001)
,
partition lored100 values less than (3750000001)
,
partition lored101 values less than (3787500001)
,
partition lored102 values less than (3825000001)
,
partition lored103 values less than (3862500001)
,
partition lored104 values less than (3900000001)
,
partition lored105 values less than (3937500001)
,
partition lored106 values less than (3975000001)
,
partition lored107 values less than (4012500001)
,
partition lored108 values less than (4050000001)
,
partition lored109 values less than (4087500001)
,
partition lored110 values less than (4125000001)
,
partition lored111 values less than (4162500001)
,
partition lored112 values less than (4200000001)
,
partition lored113 values less than (4237500001)
,
partition lored114 values less than (4275000001)
,
partition lored115 values less than (4312500001)
,
partition lored116 values less than (4350000001)
,
partition lored117 values less than (4387500001)
,
partition lored118 values less than (4425000001)
,
partition lored119 values less than (4462500001)
,
partition lored120 values less than (4500000001)
,
partition lored121 values less than (4537500001)
,
partition lored122 values less than (4575000001)
,
partition lored123 values less than (4612500001)
,
partition lored124 values less than (4650000001)
,
partition lored125 values less than (4687500001)
,
partition lored126 values less than (4725000001)
,
partition lored127 values less than (4762500001)
,
partition lored128 values less than (4800000001)
,
partition lored129 values less than (4837500001)
,
partition lored130 values less than (4875000001)
,
partition lored131 values less than (4912500001)
,
partition lored132 values less than (4950000001)
,
partition lored133 values less than (4987500001)
,
partition lored134 values less than (5025000001)
,
partition lored135 values less than (5062500001)
,
partition lored136 values less than (5100000001)
,
partition lored137 values less than (5137500001)
,
partition lored138 values less than (5175000001)
,
partition lored139 values less than (5212500001)
,
partition lored140 values less than (5250000001)
,
partition lored141 values less than (5287500001)
,
partition lored142 values less than (5325000001)
,
partition lored143 values less than (5362500001)
,
partition lored144 values less than (5400000001)
,
partition lored145 values less than (5437500001)
,
partition lored146 values less than (5475000001)
,
partition lored147 values less than (5512500001)
,
partition lored148 values less than (5550000001)
,
partition lored149 values less than (5587500001)
,
partition lored150 values less than (5625000001)
,
partition lored151 values less than (5662500001)
,
partition lored152 values less than (5700000001)
,
partition lored153 values less than (5737500001)
,
partition lored154 values less than (5775000001)
,
partition lored155 values less than (5812500001)
,
partition lored156 values less than (5850000001)
,
partition lored157 values less than (5887500001)
,
partition lored158 values less than (5925000001)
,
partition lored159 values less than (5962500001)
,
partition lored160 values less than (MAXVALUE)
)
;
}
}
*wait
*sql
{
connect tpcd/tpcd;
drop index o_clokod;
create index o_clokod
on orders (o_clerk,o_orderkey,o_orderdate)
pctfree 2
initrans 10
nologging
compute statistics
tablespace ts_ind_oclokod
storage (initial 50m next 50m freelists 32 freelist groups 5 maxextents unlimited
pctincrease 0)
parallel (degree 160 )
local
;
}
*sql
{
connect tpcd/tpcd;
drop index o_oke;
create unique index o_oke
on orders (o_orderkey)
pctfree 2
initrans 10
nologging
compute statistics
tablespace ts_ind_oke;
storage (initial 50m next 50m freelists 32 freelist groups 5 maxextents unlimited
pctincrease 0)
parallel (degree 160 )
;
}

```



```

partition ij_l_odate103 values less than (to_date('1996-04-16','YYYY-MM-DD'))
partition ij_l_odate104 values less than (to_date('1996-05-01','YYYY-MM-DD'))
partition ij_l_odate105 values less than (to_date('1996-05-16','YYYY-MM-DD'))
partition ij_l_odate106 values less than (to_date('1996-06-01','YYYY-MM-DD'))
partition ij_l_odate107 values less than (to_date('1996-06-16','YYYY-MM-DD'))
partition ij_l_odate108 values less than (to_date('1996-07-01','YYYY-MM-DD'))
partition ij_l_odate109 values less than (to_date('1996-07-16','YYYY-MM-DD'))
partition ij_l_odate110 values less than (to_date('1996-08-01','YYYY-MM-DD'))
partition ij_l_odate111 values less than (to_date('1996-08-16','YYYY-MM-DD'))
partition ij_l_odate112 values less than (to_date('1996-09-01','YYYY-MM-DD'))
partition ij_l_odate113 values less than (to_date('1996-09-16','YYYY-MM-DD'))
partition ij_l_odate114 values less than (to_date('1996-10-01','YYYY-MM-DD'))
partition ij_l_odate115 values less than (to_date('1996-10-16','YYYY-MM-DD'))
partition ij_l_odate116 values less than (to_date('1996-11-01','YYYY-MM-DD'))
partition ij_l_odate117 values less than (to_date('1996-11-16','YYYY-MM-DD'))
partition ij_l_odate118 values less than (to_date('1996-12-01','YYYY-MM-DD'))
partition ij_l_odate119 values less than (to_date('1996-12-16','YYYY-MM-DD'))
partition ij_l_odate120 values less than (to_date('1997-01-01','YYYY-MM-DD'))
partition ij_l_odate121 values less than (to_date('1997-01-16','YYYY-MM-DD'))
partition ij_l_odate122 values less than (to_date('1997-02-01','YYYY-MM-DD'))
partition ij_l_odate123 values less than (to_date('1997-02-16','YYYY-MM-DD'))
partition ij_l_odate124 values less than (to_date('1997-03-01','YYYY-MM-DD'))
partition ij_l_odate125 values less than (to_date('1997-03-16','YYYY-MM-DD'))
partition ij_l_odate126 values less than (to_date('1997-04-01','YYYY-MM-DD'))
partition ij_l_odate127 values less than (to_date('1997-04-16','YYYY-MM-DD'))
partition ij_l_odate128 values less than (to_date('1997-05-01','YYYY-MM-DD'))
partition ij_l_odate129 values less than (to_date('1997-05-16','YYYY-MM-DD'))
partition ij_l_odate130 values less than (to_date('1997-06-01','YYYY-MM-DD'))
partition ij_l_odate131 values less than (to_date('1997-06-16','YYYY-MM-DD'))
partition ij_l_odate132 values less than (to_date('1997-07-01','YYYY-MM-DD'))
partition ij_l_odate133 values less than (to_date('1997-07-16','YYYY-MM-DD'))
partition ij_l_odate134 values less than (to_date('1997-08-01','YYYY-MM-DD'))
partition ij_l_odate135 values less than (to_date('1997-08-16','YYYY-MM-DD'))
partition ij_l_odate136 values less than (to_date('1997-09-01','YYYY-MM-DD'))
partition ij_l_odate137 values less than (to_date('1997-09-16','YYYY-MM-DD'))
partition ij_l_odate138 values less than (to_date('1997-10-01','YYYY-MM-DD'))
partition ij_l_odate139 values less than (to_date('1997-10-16','YYYY-MM-DD'))
partition ij_l_odate140 values less than (to_date('1997-11-01','YYYY-MM-DD'))
partition ij_l_odate141 values less than (to_date('1997-11-16','YYYY-MM-DD'))
partition ij_l_odate142 values less than (to_date('1997-12-01','YYYY-MM-DD'))
partition ij_l_odate143 values less than (to_date('1997-12-16','YYYY-MM-DD'))
partition ij_l_odate144 values less than (to_date('1998-01-01','YYYY-MM-DD'))
partition ij_l_odate145 values less than (to_date('1998-01-16','YYYY-MM-DD'))
partition ij_l_odate146 values less than (to_date('1998-02-01','YYYY-MM-DD'))
partition ij_l_odate147 values less than (to_date('1998-02-16','YYYY-MM-DD'))
partition ij_l_odate148 values less than (to_date('1998-03-01','YYYY-MM-DD'))
partition ij_l_odate149 values less than (to_date('1998-03-16','YYYY-MM-DD'))
partition ij_l_odate150 values less than (to_date('1998-04-01','YYYY-MM-DD'))
partition ij_l_odate151 values less than (to_date('1998-04-16','YYYY-MM-DD'))
partition ij_l_odate152 values less than (to_date('1998-05-01','YYYY-MM-DD'))
partition ij_l_odate153 values less than (to_date('1998-05-16','YYYY-MM-DD'))
partition ij_l_odate154 values less than (to_date('1998-06-01','YYYY-MM-DD'))
partition ij_l_odate155 values less than (to_date('1998-06-16','YYYY-MM-DD'))
partition ij_l_odate156 values less than (to_date('1998-07-01','YYYY-MM-DD'))
partition ij_l_odate157 values less than (to_date('1998-07-16','YYYY-MM-DD'))
partition ij_l_odate158 values less than (to_date('1998-08-01','YYYY-MM-DD'))
partition ij_l_odate159 values less than (to_date('1998-08-16','YYYY-MM-DD'))
partition ij_l_odate160 values less than (MAXVALUE)
)
;
*wait
*sql
{
connect tpcd/tpcd;
drop index ij_l_sdate;
create index ij_l_sdate
on mj_plocs
(o_orderkey,o_orderdate,o_ajm,c_ajm,l_extendedprice,l_discount,c_mktsegment,l_ship
date,l_orderkey,l_partkey,o_shippriority)
pctfree 2
initrans 10
nologging
compute statistics
tablespace ts_ij_l_sdate
storage (initial 50m next 100m freelists 32 freelist groups 5 maxextents unlimited
pctincrease 0)
parallel (degree 160 )
local
;
}
*wait
*sql
{
connect tpcd/tpcd;
drop index ij_l_rdate;
create index ij_l_rdate
on mj_plocs
(l_receiptdate,o_orderkey,o_ajm,l_shipdate,l_shipmode,l_commitdate,o_orderpriority)
pctfree 2
initrans 10
nologging
compute statistics
tablespace ts_ij_l_rdate
storage (initial 100m next 100m freelists 32 freelist groups 5 maxextents
unlimited pctincrease 0)
parallel (degree 160 )
global partition by range (l_receiptdate)
(
partition ij_l_rdate1 values less than (to_date('1992-03-01','YYYY-MM-DD'))
partition ij_l_rdate2 values less than (to_date('1992-03-24','YYYY-MM-DD'))
partition ij_l_rdate3 values less than (to_date('1992-04-14','YYYY-MM-DD'))
partition ij_l_rdate4 values less than (to_date('1992-05-01','YYYY-MM-DD'))
partition ij_l_rdate5 values less than (to_date('1992-05-16','YYYY-MM-DD'))
partition ij_l_rdate6 values less than (to_date('1992-06-01','YYYY-MM-DD'))
partition ij_l_rdate7 values less than (to_date('1992-06-16','YYYY-MM-DD'))
partition ij_l_rdate8 values less than (to_date('1992-07-01','YYYY-MM-DD'))
partition ij_l_rdate9 values less than (to_date('1992-07-16','YYYY-MM-DD'))
partition ij_l_rdate10 values less than (to_date('1992-08-01','YYYY-MM-DD'))
partition ij_l_rdate11 values less than (to_date('1992-08-16','YYYY-MM-DD'))
partition ij_l_rdate12 values less than (to_date('1992-09-01','YYYY-MM-DD'))
partition ij_l_rdate13 values less than (to_date('1992-09-16','YYYY-MM-DD'))
partition ij_l_rdate14 values less than (to_date('1992-10-01','YYYY-MM-DD'))
partition ij_l_rdate15 values less than (to_date('1992-10-16','YYYY-MM-DD'))
partition ij_l_rdate16 values less than (to_date('1992-11-01','YYYY-MM-DD'))
partition ij_l_rdate17 values less than (to_date('1992-11-16','YYYY-MM-DD'))
partition ij_l_rdate18 values less than (to_date('1992-12-01','YYYY-MM-DD'))
partition ij_l_rdate19 values less than (to_date('1992-12-16','YYYY-MM-DD'))
partition ij_l_rdate20 values less than (to_date('1993-01-01','YYYY-MM-DD'))
partition ij_l_rdate21 values less than (to_date('1993-01-16','YYYY-MM-DD'))
partition ij_l_rdate22 values less than (to_date('1993-02-01','YYYY-MM-DD'))
partition ij_l_rdate23 values less than (to_date('1993-02-16','YYYY-MM-DD'))
partition ij_l_rdate24 values less than (to_date('1993-03-01','YYYY-MM-DD'))
partition ij_l_rdate25 values less than (to_date('1993-03-16','YYYY-MM-DD'))
partition ij_l_rdate26 values less than (to_date('1993-04-01','YYYY-MM-DD'))
partition ij_l_rdate27 values less than (to_date('1993-04-16','YYYY-MM-DD'))
partition ij_l_rdate28 values less than (to_date('1993-05-01','YYYY-MM-DD'))
partition ij_l_rdate29 values less than (to_date('1993-05-16','YYYY-MM-DD'))
partition ij_l_rdate30 values less than (to_date('1993-06-01','YYYY-MM-DD'))
partition ij_l_rdate31 values less than (to_date('1993-06-16','YYYY-MM-DD'))
partition ij_l_rdate32 values less than (to_date('1993-07-01','YYYY-MM-DD'))
partition ij_l_rdate33 values less than (to_date('1993-07-16','YYYY-MM-DD'))
)
;

```

[illegible][illegible]

```

partition ij_l_rdate145 values less than (to_date('1998-03-16','YYYY-MM-DD'))
,partition ij_l_rdate146 values less than (to_date('1998-04-01','YYYY-MM-DD'))
,partition ij_l_rdate147 values less than (to_date('1998-04-16','YYYY-MM-DD'))
,partition ij_l_rdate148 values less than (to_date('1998-05-01','YYYY-MM-DD'))
,partition ij_l_rdate149 values less than (to_date('1998-05-16','YYYY-MM-DD'))
,partition ij_l_rdate150 values less than (to_date('1998-06-01','YYYY-MM-DD'))
,partition ij_l_rdate151 values less than (to_date('1998-06-16','YYYY-MM-DD'))
,partition ij_l_rdate152 values less than (to_date('1998-07-01','YYYY-MM-DD'))
,partition ij_l_rdate153 values less than (to_date('1998-07-16','YYYY-MM-DD'))
,partition ij_l_rdate154 values less than (to_date('1998-08-01','YYYY-MM-DD'))
,partition ij_l_rdate155 values less than (to_date('1998-08-16','YYYY-MM-DD'))
,partition ij_l_rdate156 values less than (to_date('1998-09-01','YYYY-MM-DD'))
,partition ij_l_rdate157 values less than (to_date('1998-09-16','YYYY-MM-DD'))
,partition ij_l_rdate158 values less than (to_date('1998-10-01','YYYY-MM-DD'))
,partition ij_l_rdate159 values less than (to_date('1998-11-01','YYYY-MM-DD'))
,partition ij_l_rdate160 values less than (MAXVALUE)
)
;
*wait
*sql
{
connect tpcd/tpcd;
drop index ij_l_pkey;
create index ij_l_pkey
on mj_plocs
(l_partkey,o_orderkey,l_extendedprice,l_discount,l_quantity,o_orderdate,o_ajm,c_aj
m,s_ajm,l_suppkkey,s_nationkey,p_ajm,p_rowid)
pctfree 2
initrans 10
nologging
compute statistics
tablespace ts_ij_l_pkey
storage (initial 100m next 100m freelists 32 freelist groups 5 maxextents
unlimited pctincrease 0)
parallel (degree 160 )
global partition by range (l_partkey)
(
partition ij_l_pkey1 values less than (1250001)
,partition ij_l_pkey2 values less than (2500001)
,partition ij_l_pkey3 values less than (3750001)
,partition ij_l_pkey4 values less than (5000001)
,partition ij_l_pkey5 values less than (6250001)
,partition ij_l_pkey6 values less than (7500001)
,partition ij_l_pkey7 values less than (8750001)
,partition ij_l_pkey8 values less than (10000001)
,partition ij_l_pkey9 values less than (11250001)
,partition ij_l_pkey10 values less than (12500001)
,partition ij_l_pkey11 values less than (13750001)
,partition ij_l_pkey12 values less than (15000001)
,partition ij_l_pkey13 values less than (16250001)
,partition ij_l_pkey14 values less than (17500001)
,partition ij_l_pkey15 values less than (18750001)
,partition ij_l_pkey16 values less than (20000001)
,partition ij_l_pkey17 values less than (21250001)
,partition ij_l_pkey18 values less than (22500001)
,partition ij_l_pkey19 values less than (23750001)
,partition ij_l_pkey20 values less than (25000001)
,partition ij_l_pkey21 values less than (26250001)
,partition ij_l_pkey22 values less than (27500001)
,partition ij_l_pkey23 values less than (28750001)
,partition ij_l_pkey24 values less than (30000001)
,partition ij_l_pkey25 values less than (31250001)
,partition ij_l_pkey26 values less than (32500001)
,partition ij_l_pkey27 values less than (33750001)
,partition ij_l_pkey28 values less than (35000001)
,partition ij_l_pkey29 values less than (36250001)
,partition ij_l_pkey30 values less than (37500001)
,partition ij_l_pkey31 values less than (38750001)
,partition ij_l_pkey32 values less than (40000001)
,partition ij_l_pkey33 values less than (41250001)
,partition ij_l_pkey34 values less than (42500001)
,partition ij_l_pkey35 values less than (43750001)
,partition ij_l_pkey36 values less than (45000001)
,partition ij_l_pkey37 values less than (46250001)
,partition ij_l_pkey38 values less than (47500001)
,partition ij_l_pkey39 values less than (48750001)
,partition ij_l_pkey40 values less than (50000001)
,partition ij_l_pkey41 values less than (51250001)
,partition ij_l_pkey42 values less than (52500001)
,partition ij_l_pkey43 values less than (53750001)
,partition ij_l_pkey44 values less than (55000001)
,partition ij_l_pkey45 values less than (56250001)
,partition ij_l_pkey46 values less than (57500001)
,partition ij_l_pkey47 values less than (58750001)
,partition ij_l_pkey48 values less than (60000001)
,partition ij_l_pkey49 values less than (61250001)
,partition ij_l_pkey50 values less than (62500001)
,partition ij_l_pkey51 values less than (63750001)
,partition ij_l_pkey52 values less than (65000001)
,partition ij_l_pkey53 values less than (66250001)
,partition ij_l_pkey54 values less than (67500001)
,partition ij_l_pkey55 values less than (68750001)
,partition ij_l_pkey56 values less than (70000001)
,partition ij_l_pkey57 values less than (71250001)
,partition ij_l_pkey58 values less than (72500001)
,partition ij_l_pkey59 values less than (73750001)
,partition ij_l_pkey60 values less than (75000001)
,partition ij_l_pkey61 values less than (76250001)
,partition ij_l_pkey62 values less than (77500001)
,partition ij_l_pkey63 values less than (78750001)
,partition ij_l_pkey64 values less than (80000001)
,partition ij_l_pkey65 values less than (81250001)
,partition ij_l_pkey66 values less than (82500001)
,partition ij_l_pkey67 values less than (83750001)
,partition ij_l_pkey68 values less than (85000001)
,partition ij_l_pkey69 values less than (86250001)
,partition ij_l_pkey70 values less than (87500001)
,partition ij_l_pkey71 values less than (88750001)
,partition ij_l_pkey72 values less than (90000001)
,partition ij_l_pkey73 values less than (91250001)
,partition ij_l_pkey74 values less than (92500001)
,partition ij_l_pkey75 values less than (93750001)
,partition ij_l_pkey76 values less than (95000001)
,partition ij_l_pkey77 values less than (96250001)
,partition ij_l_pkey78 values less than (97500001)
,partition ij_l_pkey79 values less than (98750001)
,partition ij_l_pkey80 values less than (100000001)
,partition ij_l_pkey81 values less than (101250001)
,partition ij_l_pkey82 values less than (102500001)
,partition ij_l_pkey83 values less than (103750001)
,partition ij_l_pkey84 values less than (105000001)
,partition ij_l_pkey85 values less than (106250001)
,

```

```

partition ij_l_pkey86 values less than (107500001)
partition ij_l_pkey87 values less than (108750001)
partition ij_l_pkey88 values less than (110000001)
partition ij_l_pkey89 values less than (111250001)
partition ij_l_pkey90 values less than (112500001)
partition ij_l_pkey91 values less than (113750001)
partition ij_l_pkey92 values less than (115000001)
partition ij_l_pkey93 values less than (116250001)
partition ij_l_pkey94 values less than (117500001)
partition ij_l_pkey95 values less than (118750001)
partition ij_l_pkey96 values less than (120000001)
partition ij_l_pkey97 values less than (121250001)
partition ij_l_pkey98 values less than (122500001)
partition ij_l_pkey99 values less than (123750001)
partition ij_l_pkey100 values less than (125000001)
partition ij_l_pkey101 values less than (126250001)
partition ij_l_pkey102 values less than (127500001)
partition ij_l_pkey103 values less than (128750001)
partition ij_l_pkey104 values less than (130000001)
partition ij_l_pkey105 values less than (131250001)
partition ij_l_pkey106 values less than (132500001)
partition ij_l_pkey107 values less than (133750001)
partition ij_l_pkey108 values less than (135000001)
partition ij_l_pkey109 values less than (136250001)
partition ij_l_pkey110 values less than (137500001)
partition ij_l_pkey111 values less than (138750001)
partition ij_l_pkey112 values less than (140000001)
partition ij_l_pkey113 values less than (141250001)
partition ij_l_pkey114 values less than (142500001)
partition ij_l_pkey115 values less than (143750001)
partition ij_l_pkey116 values less than (145000001)
partition ij_l_pkey117 values less than (146250001)
partition ij_l_pkey118 values less than (147500001)
partition ij_l_pkey119 values less than (148750001)
partition ij_l_pkey120 values less than (150000001)
partition ij_l_pkey121 values less than (151250001)
partition ij_l_pkey122 values less than (152500001)
partition ij_l_pkey123 values less than (153750001)
partition ij_l_pkey124 values less than (155000001)
partition ij_l_pkey125 values less than (156250001)
partition ij_l_pkey126 values less than (157500001)
partition ij_l_pkey127 values less than (158750001)
partition ij_l_pkey128 values less than (160000001)
partition ij_l_pkey129 values less than (161250001)
partition ij_l_pkey130 values less than (162500001)
partition ij_l_pkey131 values less than (163750001)
partition ij_l_pkey132 values less than (165000001)
partition ij_l_pkey133 values less than (166250001)
partition ij_l_pkey134 values less than (167500001)
partition ij_l_pkey135 values less than (168750001)
partition ij_l_pkey136 values less than (170000001)
partition ij_l_pkey137 values less than (171250001)
partition ij_l_pkey138 values less than (172500001)
partition ij_l_pkey139 values less than (173750001)
partition ij_l_pkey140 values less than (175000001)
partition ij_l_pkey141 values less than (176250001)

partition ij_l_pkey142 values less than (177500001)
partition ij_l_pkey143 values less than (178750001)
partition ij_l_pkey144 values less than (180000001)
partition ij_l_pkey145 values less than (181250001)
partition ij_l_pkey146 values less than (182500001)
partition ij_l_pkey147 values less than (183750001)
partition ij_l_pkey148 values less than (185000001)
partition ij_l_pkey149 values less than (186250001)
partition ij_l_pkey150 values less than (187500001)
partition ij_l_pkey151 values less than (188750001)
partition ij_l_pkey152 values less than (190000001)
partition ij_l_pkey153 values less than (191250001)
partition ij_l_pkey154 values less than (192500001)
partition ij_l_pkey155 values less than (193750001)
partition ij_l_pkey156 values less than (195000001)
partition ij_l_pkey157 values less than (196250001)
partition ij_l_pkey158 values less than (197500001)
partition ij_l_pkey159 values less than (198750001)
partition ij_l_pkey160 values less than (MAXVALUE)
)
;
}
*wait
*sql
{
connect tpcd/tpcd;
drop index ij_l_lrid;
create index ij_l_lrid
on mj_plocs (l_rowid)
pctfree 2
initrans 10
nologging
compute statistics
tablespace ts_ij_l_lrid
storage (initial 50m next 50m freelists 32 freelist groups 5 maxextents unlimited
pctincrease 0)
parallel (degree 160)
;
}
*wait
*sql
{
connect tpcd/tpcd;
drop index ij_l_orid;
create index ij_l_orid
on mj_plocs (o_rowid)
pctfree 2
initrans 10
nologging
compute statistics
tablespace ts_ij_l_orid
storage (initial 50m next 50m freelists 32 freelist groups 5 maxextents unlimited
pctincrease 0)
parallel (degree 160)
;
}
*wait
*sql
{
connect tpcd/tpcd;
drop index ij_l_nullrid;
create index ij_l_nullrid
on mj_plocs (decode(o_rowid, NULL, 1, NULL))
pctfree 2
initrans 10
nologging
compute statistics
tablespace ts_ij_l_nullrid
storage (initial 10m next 10m freelists 32 freelist groups 5 maxextents unlimited
pctincrease 0)
parallel (degree 160 )
;
}
*wait
*sql
{
connect tpcd/tpcd;
drop index ij_p_ppsn;
create index ij_p_ppsn
on mj_ppssrn (p_partkey,ps_partkey,ps_supplycost,n_ajm,r_ajm,r_name,s_ajm)
pctfree 0
nologging
compute statistics
tablespace ts_ij_p_ppsn
storage (initial 10m next 20m maxextents unlimited pctincrease 0)
parallel (degree 160)
local
;
}
*wait
*sql
{
connect tpcd/tpcd;

```

```

drop index ij_p_nnps;
create index ij_p_nnps
on mj_ppssrn
(
    ps_partkey,
    p_partkey,
    p_name,
    n_name,
    n_ajm,
    p_ajm,
    s_ajm,
    ps_suppkey,
    ps_supplycost,
    s_suppkey
)
pctfree 0
nologging
compute statistics
tablespace ts_ij_p_nnps
storage (initial 50m next 50m maxextents unlimited pctincrease 0)
parallel (degree 160)
global partition by range (ps_partkey)
(
    partition ij_p_nnps1 values less than (1250001)
    ,
    partition ij_p_nnps2 values less than (2500001)
    ,
    partition ij_p_nnps3 values less than (3750001)
    ,
    partition ij_p_nnps4 values less than (5000001)
    ,
    partition ij_p_nnps5 values less than (6250001)
    ,
    partition ij_p_nnps6 values less than (7500001)
    ,
    partition ij_p_nnps7 values less than (8750001)
    ,
    partition ij_p_nnps8 values less than (10000001)
    ,
    partition ij_p_nnps9 values less than (11250001)
    ,
    partition ij_p_nnps10 values less than (12500001)
    ,
    partition ij_p_nnps11 values less than (13750001)
    ,
    partition ij_p_nnps12 values less than (15000001)
    ,
    partition ij_p_nnps13 values less than (16250001)
    ,
    partition ij_p_nnps14 values less than (17500001)
    ,
    partition ij_p_nnps15 values less than (18750001)
    ,
    partition ij_p_nnps16 values less than (20000001)
    ,
    partition ij_p_nnps17 values less than (21250001)
    ,
    partition ij_p_nnps18 values less than (22500001)
    ,
    partition ij_p_nnps19 values less than (23750001)
    ,
    partition ij_p_nnps20 values less than (25000001)
    ,
    partition ij_p_nnps21 values less than (26250001)
    ,
    partition ij_p_nnps22 values less than (27500001)
    ,
    partition ij_p_nnps23 values less than (28750001)
    ,
    partition ij_p_nnps24 values less than (30000001)
    ,
    partition ij_p_nnps25 values less than (31250001)
    ,
    partition ij_p_nnps26 values less than (32500001)
    ,
    partition ij_p_nnps27 values less than (33750001)
    ,
    partition ij_p_nnps28 values less than (35000001)
    ,
    partition ij_p_nnps29 values less than (36250001)
    ,
    partition ij_p_nnps30 values less than (37500001)
    ,
    partition ij_p_nnps31 values less than (38750001)
    ,
    partition ij_p_nnps32 values less than (40000001)
    ,
    partition ij_p_nnps33 values less than (41250001)
    ,
    partition ij_p_nnps34 values less than (42500001)
    ,
    partition ij_p_nnps35 values less than (43750001)
    ,
    partition ij_p_nnps36 values less than (45000001)
    ,
    partition ij_p_nnps37 values less than (46250001)
    ,
    partition ij_p_nnps38 values less than (47500001)
    ,
    partition ij_p_nnps39 values less than (48750001)
    ,
    partition ij_p_nnps40 values less than (50000001)
    ,
    partition ij_p_nnps41 values less than (51250001)
    ,
    partition ij_p_nnps42 values less than (52500001)
    ,
    partition ij_p_nnps43 values less than (53750001)
    ,
    partition ij_p_nnps44 values less than (55000001)
    ,

```

```

    partition ij_p_nnps45 values less than (56250001)
    ,
    partition ij_p_nnps46 values less than (57500001)
    ,
    partition ij_p_nnps47 values less than (58750001)
    ,
    partition ij_p_nnps48 values less than (60000001)
    ,
    partition ij_p_nnps49 values less than (61250001)
    ,
    partition ij_p_nnps50 values less than (62500001)
    ,
    partition ij_p_nnps51 values less than (63750001)
    ,
    partition ij_p_nnps52 values less than (65000001)
    ,
    partition ij_p_nnps53 values less than (66250001)
    ,
    partition ij_p_nnps54 values less than (67500001)
    ,
    partition ij_p_nnps55 values less than (68750001)
    ,
    partition ij_p_nnps56 values less than (70000001)
    ,
    partition ij_p_nnps57 values less than (71250001)
    ,
    partition ij_p_nnps58 values less than (72500001)
    ,
    partition ij_p_nnps59 values less than (73750001)
    ,
    partition ij_p_nnps60 values less than (75000001)
    ,
    partition ij_p_nnps61 values less than (76250001)
    ,
    partition ij_p_nnps62 values less than (77500001)
    ,
    partition ij_p_nnps63 values less than (78750001)
    ,
    partition ij_p_nnps64 values less than (80000001)
    ,
    partition ij_p_nnps65 values less than (81250001)
    ,
    partition ij_p_nnps66 values less than (82500001)
    ,
    partition ij_p_nnps67 values less than (83750001)
    ,
    partition ij_p_nnps68 values less than (85000001)
    ,
    partition ij_p_nnps69 values less than (86250001)
    ,
    partition ij_p_nnps70 values less than (87500001)
    ,
    partition ij_p_nnps71 values less than (88750001)
    ,
    partition ij_p_nnps72 values less than (90000001)
    ,
    partition ij_p_nnps73 values less than (91250001)
    ,
    partition ij_p_nnps74 values less than (92500001)
    ,
    partition ij_p_nnps75 values less than (93750001)
    ,
    partition ij_p_nnps76 values less than (95000001)
    ,
    partition ij_p_nnps77 values less than (96250001)
    ,
    partition ij_p_nnps78 values less than (97500001)
    ,
    partition ij_p_nnps79 values less than (98750001)
    ,
    partition ij_p_nnps80 values less than (100000001)
    ,
    partition ij_p_nnps81 values less than (101250001)
    ,
    partition ij_p_nnps82 values less than (102500001)
    ,
    partition ij_p_nnps83 values less than (103750001)
    ,
    partition ij_p_nnps84 values less than (105000001)
    ,
    partition ij_p_nnps85 values less than (106250001)
    ,
    partition ij_p_nnps86 values less than (107500001)
    ,
    partition ij_p_nnps87 values less than (108750001)
    ,
    partition ij_p_nnps88 values less than (110000001)
    ,
    partition ij_p_nnps89 values less than (111250001)
    ,
    partition ij_p_nnps90 values less than (112500001)
    ,
    partition ij_p_nnps91 values less than (113750001)
    ,
    partition ij_p_nnps92 values less than (115000001)
    ,
    partition ij_p_nnps93 values less than (116250001)
    ,
    partition ij_p_nnps94 values less than (117500001)
    ,
    partition ij_p_nnps95 values less than (118750001)
    ,
    partition ij_p_nnps96 values less than (120000001)
    ,
    partition ij_p_nnps97 values less than (121250001)
    ,
    partition ij_p_nnps98 values less than (122500001)
    ,
    partition ij_p_nnps99 values less than (123750001)
    ,
    partition ij_p_nnps100 values less than (125000001)

```

```

,
partition ij_p_nnps101 values less than (126250001)
,
partition ij_p_nnps102 values less than (127500001)
,
partition ij_p_nnps103 values less than (128750001)
,
partition ij_p_nnps104 values less than (130000001)
,
partition ij_p_nnps105 values less than (131250001)
,
partition ij_p_nnps106 values less than (132500001)
,
partition ij_p_nnps107 values less than (133750001)
,
partition ij_p_nnps108 values less than (135000001)
,
partition ij_p_nnps109 values less than (136250001)
,
partition ij_p_nnps110 values less than (137500001)
,
partition ij_p_nnps111 values less than (138750001)
,
partition ij_p_nnps112 values less than (140000001)
,
partition ij_p_nnps113 values less than (141250001)
,
partition ij_p_nnps114 values less than (142500001)
,
partition ij_p_nnps115 values less than (143750001)
,
partition ij_p_nnps116 values less than (145000001)
,
partition ij_p_nnps117 values less than (146250001)
,
partition ij_p_nnps118 values less than (147500001)
,
partition ij_p_nnps119 values less than (148750001)
,
partition ij_p_nnps120 values less than (150000001)
,
partition ij_p_nnps121 values less than (151250001)
,
partition ij_p_nnps122 values less than (152500001)
,
partition ij_p_nnps123 values less than (153750001)
,
partition ij_p_nnps124 values less than (155000001)
,
partition ij_p_nnps125 values less than (156250001)
,
partition ij_p_nnps126 values less than (157500001)
,
partition ij_p_nnps127 values less than (158750001)
,
partition ij_p_nnps128 values less than (160000001)
,
partition ij_p_nnps129 values less than (161250001)
,
partition ij_p_nnps130 values less than (162500001)
,
partition ij_p_nnps131 values less than (163750001)
,
partition ij_p_nnps132 values less than (165000001)
,
partition ij_p_nnps133 values less than (166250001)
,
partition ij_p_nnps134 values less than (167500001)
,
partition ij_p_nnps135 values less than (168750001)
,
partition ij_p_nnps136 values less than (170000001)
,
partition ij_p_nnps137 values less than (171250001)
,
partition ij_p_nnps138 values less than (172500001)
,
partition ij_p_nnps139 values less than (173750001)
,
partition ij_p_nnps140 values less than (175000001)
,
partition ij_p_nnps141 values less than (176250001)
,
partition ij_p_nnps142 values less than (177500001)
,
partition ij_p_nnps143 values less than (178750001)
,
partition ij_p_nnps144 values less than (180000001)
,
partition ij_p_nnps145 values less than (181250001)
,
partition ij_p_nnps146 values less than (182500001)
,
partition ij_p_nnps147 values less than (183750001)
,
partition ij_p_nnps148 values less than (185000001)
,
partition ij_p_nnps149 values less than (186250001)
,
partition ij_p_nnps150 values less than (187500001)
,
partition ij_p_nnps151 values less than (188750001)
,
partition ij_p_nnps152 values less than (190000001)
,
partition ij_p_nnps153 values less than (191250001)
,
partition ij_p_nnps154 values less than (192500001)
,
partition ij_p_nnps155 values less than (193750001)
,
,
partition ij_p_nnps156 values less than (195000001)
,
partition ij_p_nnps157 values less than (196250001)
,
partition ij_p_nnps158 values less than (197500001)
,
partition ij_p_nnps159 values less than (198750001)
,
partition ij_p_nnps160 values less than (MAXVALUE)
)
)
;
;
*wait
*sql
{
connect tpcd/tpcd;
drop index ij_p_nnpsps;
create index ij_p_nnpsps
on mj_ppssrn (n_name,n_ajm,s_ajm,ps_partkey,ps_supplycost,ps_availqty)
pctfree 0
nologging
compute statistics
tablespace ts_ij_p_nnpsps
storage (initial 50m next 50m maxextents unlimited pctincrease 0)
parallel
global partition by range (n_name)
(
partition ij_p_nnpsps1 values less than ('ALGERIB')
,
partition ij_p_nnpsps2 values less than ('ARGENTINB')
,
partition ij_p_nnpsps3 values less than ('BRAZIM')
,
partition ij_p_nnpsps4 values less than ('CANADB')
,
partition ij_p_nnpsps5 values less than ('CHINB')
,
partition ij_p_nnpsps6 values less than ('EGYPY')
,
partition ij_p_nnpsps7 values less than ('ETHIOPIB')
,
partition ij_p_nnpsps8 values less than ('FRANCF')
,
partition ij_p_nnpsps9 values less than ('GERMANZ')
,
partition ij_p_nnpsps10 values less than ('INDIB')
,
partition ij_p_nnpsps11 values less than ('INDONESIB')
,
partition ij_p_nnpsps12 values less than ('IRAQ')
,
partition ij_p_nnpsps13 values less than ('IRAR')
,
partition ij_p_nnpsps14 values less than ('JAPAO')
,
partition ij_p_nnpsps15 values less than ('JORDAO')
,
partition ij_p_nnpsps16 values less than ('KENYB')
,
partition ij_p_nnpsps17 values less than ('MOROCCP')
,
partition ij_p_nnpsps18 values less than ('MOZAMBIQUF')
,
partition ij_p_nnpsps19 values less than ('PERV')
,
partition ij_p_nnpsps20 values less than ('ROMANIB')
,
partition ij_p_nnpsps21 values less than ('RUSSIB')
,
partition ij_p_nnpsps22 values less than ('SAUDI ARABIB')
,
partition ij_p_nnpsps23 values less than ('UNITED KINGDON')
,
partition ij_p_nnpsps24 values less than ('UNITED STATET')
,
partition ij_p_nnpsps25 values less than (MAXVALUE)
)
)
;
;
*sql
{
connect tpcd/tpcd;
drop index ij_l_ptp;
create index ij_l_ptp on mj_plocs
(
p_type,
o_orderkey,
l_extendedprice,
l_discount,
o_orderdate,
o_ajm,
c_ajm,
s_ajm,
p_ajm,
c_nationkey,
s_nationkey,
l_partkey
)
pctfree 2
intrans 10
nologging
compute statistics
tablespace ts_ij_l_ptp
storage (initial 50m next 50m freelists 32 freelist groups 5 maxextents unlimited
pctincrease 0)
parallel (degree 160)
global partition by range (p_type)
(
partition ptpl values less than ('ECONOMY ANODIZED COPPER')
,
partition ptp2 values less than ('ECONOMY ANODIZED NICKEL')

```

,
partition ptp3 values less than ('ECONOMY ANODIZED STEEL')
,
partition ptp4 values less than ('ECONOMY ANODIZED TIN')
,
partition ptp5 values less than ('ECONOMY BRUSHED BRASS')
,
partition ptp6 values less than ('ECONOMY BRUSHED COPPER')
,
partition ptp7 values less than ('ECONOMY BRUSHED NICKEL')
,
partition ptp8 values less than ('ECONOMY BRUSHED STEEL')
,
partition ptp9 values less than ('ECONOMY BRUSHED TIN')
,
partition ptp10 values less than ('ECONOMY BURNISHED BRASS')
,
partition ptp11 values less than ('ECONOMY BURNISHED COPPER')
,
partition ptp12 values less than ('ECONOMY BURNISHED NICKEL')
,
partition ptp13 values less than ('ECONOMY BURNISHED STEEL')
,
partition ptp14 values less than ('ECONOMY BURNISHED TIN')
,
partition ptp15 values less than ('ECONOMY PLATED BRASS')
,
partition ptp16 values less than ('ECONOMY PLATED COPPER')
,
partition ptp17 values less than ('ECONOMY PLATED NICKEL')
,
partition ptp18 values less than ('ECONOMY PLATED STEEL')
,
partition ptp19 values less than ('ECONOMY PLATED TIN')
,
partition ptp20 values less than ('ECONOMY POLISHED BRASS')
,
partition ptp21 values less than ('ECONOMY POLISHED COPPER')
,
partition ptp22 values less than ('ECONOMY POLISHED NICKEL')
,
partition ptp23 values less than ('ECONOMY POLISHED STEEL')
,
partition ptp24 values less than ('ECONOMY POLISHED TIN')
,
partition ptp25 values less than ('LARGE ANODIZED BRASS')
,
partition ptp26 values less than ('LARGE ANODIZED COPPER')
,
partition ptp27 values less than ('LARGE ANODIZED NICKEL')
,
partition ptp28 values less than ('LARGE ANODIZED STEEL')
,
partition ptp29 values less than ('LARGE ANODIZED TIN')
,
partition ptp30 values less than ('LARGE BRUSHED BRASS')
,
partition ptp31 values less than ('LARGE BRUSHED COPPER')
,
partition ptp32 values less than ('LARGE BRUSHED NICKEL')
,
partition ptp33 values less than ('LARGE BRUSHED STEEL')
,
partition ptp34 values less than ('LARGE BRUSHED TIN')
,
partition ptp35 values less than ('LARGE BURNISHED BRASS')
,
partition ptp36 values less than ('LARGE BURNISHED COPPER')
,
partition ptp37 values less than ('LARGE BURNISHED NICKEL')
,
partition ptp38 values less than ('LARGE BURNISHED STEEL')
,
partition ptp39 values less than ('LARGE BURNISHED TIN')
,
partition ptp40 values less than ('LARGE PLATED BRASS')
,
partition ptp41 values less than ('LARGE PLATED COPPER')
,
partition ptp42 values less than ('LARGE PLATED NICKEL')
,
partition ptp43 values less than ('LARGE PLATED STEEL')
,
partition ptp44 values less than ('LARGE PLATED TIN')
,
partition ptp45 values less than ('LARGE POLISHED BRASS')
,
partition ptp46 values less than ('LARGE POLISHED COPPER')
,
partition ptp47 values less than ('LARGE POLISHED NICKEL')
,
partition ptp48 values less than ('LARGE POLISHED STEEL')
,
partition ptp49 values less than ('LARGE POLISHED TIN')
,
partition ptp50 values less than ('MEDIUM ANODIZED BRASS')
,
partition ptp51 values less than ('MEDIUM ANODIZED COPPER')
,
partition ptp52 values less than ('MEDIUM ANODIZED NICKEL')
,
partition ptp53 values less than ('MEDIUM ANODIZED STEEL')
,
partition ptp54 values less than ('MEDIUM ANODIZED TIN')
,
partition ptp55 values less than ('MEDIUM BRUSHED BRASS')
,
partition ptp56 values less than ('MEDIUM BRUSHED COPPER')
,
partition ptp57 values less than ('MEDIUM BRUSHED NICKEL')
,

partition ptp58 values less than ('MEDIUM BRUSHED STEEL')
,
partition ptp59 values less than ('MEDIUM BRUSHED TIN')
,
partition ptp60 values less than ('MEDIUM BURNISHED BRASS')
,
partition ptp61 values less than ('MEDIUM BURNISHED COPPER')
,
partition ptp62 values less than ('MEDIUM BURNISHED NICKEL')
,
partition ptp63 values less than ('MEDIUM BURNISHED STEEL')
,
partition ptp64 values less than ('MEDIUM BURNISHED TIN')
,
partition ptp65 values less than ('MEDIUM PLATED BRASS')
,
partition ptp66 values less than ('MEDIUM PLATED COPPER')
,
partition ptp67 values less than ('MEDIUM PLATED NICKEL')
,
partition ptp68 values less than ('MEDIUM PLATED STEEL')
,
partition ptp69 values less than ('MEDIUM PLATED TIN')
,
partition ptp70 values less than ('MEDIUM POLISHED BRASS')
,
partition ptp71 values less than ('MEDIUM POLISHED COPPER')
,
partition ptp72 values less than ('MEDIUM POLISHED NICKEL')
,
partition ptp73 values less than ('MEDIUM POLISHED STEEL')
,
partition ptp74 values less than ('MEDIUM POLISHED TIN')
,
partition ptp75 values less than ('PROMO ANODIZED BRASS')
,
partition ptp76 values less than ('PROMO ANODIZED COPPER')
,
partition ptp77 values less than ('PROMO ANODIZED NICKEL')
,
partition ptp78 values less than ('PROMO ANODIZED STEEL')
,
partition ptp79 values less than ('PROMO ANODIZED TIN')
,
partition ptp80 values less than ('PROMO BRUSHED BRASS')
,
partition ptp81 values less than ('PROMO BRUSHED COPPER')
,
partition ptp82 values less than ('PROMO BRUSHED NICKEL')
,
partition ptp83 values less than ('PROMO BRUSHED STEEL')
,
partition ptp84 values less than ('PROMO BRUSHED TIN')
,
partition ptp85 values less than ('PROMO BURNISHED BRASS')
,
partition ptp86 values less than ('PROMO BURNISHED COPPER')
,
partition ptp87 values less than ('PROMO BURNISHED NICKEL')
,
partition ptp88 values less than ('PROMO BURNISHED STEEL')
,
partition ptp89 values less than ('PROMO BURNISHED TIN')
,
partition ptp90 values less than ('PROMO PLATED BRASS')
,
partition ptp91 values less than ('PROMO PLATED COPPER')
,
partition ptp92 values less than ('PROMO PLATED NICKEL')
,
partition ptp93 values less than ('PROMO PLATED STEEL')
,
partition ptp94 values less than ('PROMO PLATED TIN')
,
partition ptp95 values less than ('PROMO POLISHED BRASS')
,
partition ptp96 values less than ('PROMO POLISHED COPPER')
,
partition ptp97 values less than ('PROMO POLISHED NICKEL')
,
partition ptp98 values less than ('PROMO POLISHED STEEL')
,
partition ptp99 values less than ('PROMO POLISHED TIN')
,
partition ptp100 values less than ('SMALL ANODIZED BRASS')
,
partition ptp101 values less than ('SMALL ANODIZED COPPER')
,
partition ptp102 values less than ('SMALL ANODIZED NICKEL')
,
partition ptp103 values less than ('SMALL ANODIZED STEEL')
,
partition ptp104 values less than ('SMALL ANODIZED TIN')
,
partition ptp105 values less than ('SMALL BRUSHED BRASS')
,
partition ptp106 values less than ('SMALL BRUSHED COPPER')
,
partition ptp107 values less than ('SMALL BRUSHED NICKEL')
,
partition ptp108 values less than ('SMALL BRUSHED STEEL')
,
partition ptp109 values less than ('SMALL BRUSHED TIN')
,
partition ptp110 values less than ('SMALL BURNISHED BRASS')
,
partition ptp111 values less than ('SMALL BURNISHED COPPER')
,
partition ptp112 values less than ('SMALL BURNISHED NICKEL')
,
partition ptp113 values less than ('SMALL BURNISHED STEEL')

```

partition ptp114 values less than ('SMALL BURNISHED TIN')
partition ptp115 values less than ('SMALL PLATED BRASS')
partition ptp116 values less than ('SMALL PLATED COPPER')
partition ptp117 values less than ('SMALL PLATED NICKEL')
partition ptp118 values less than ('SMALL PLATED STEEL')
partition ptp119 values less than ('SMALL PLATED TIN')
partition ptp120 values less than ('SMALL POLISHED BRASS')
partition ptp121 values less than ('SMALL POLISHED COPPER')
partition ptp122 values less than ('SMALL POLISHED NICKEL')
partition ptp123 values less than ('SMALL POLISHED STEEL')
partition ptp124 values less than ('SMALL POLISHED TIN')
partition ptp125 values less than ('STANDARD ANODIZED BRASS')
partition ptp126 values less than ('STANDARD ANODIZED COPPER')
partition ptp127 values less than ('STANDARD ANODIZED NICKEL')
partition ptp128 values less than ('STANDARD ANODIZED STEEL')
partition ptp129 values less than ('STANDARD ANODIZED TIN')
partition ptp130 values less than ('STANDARD BRUSHED BRASS')
partition ptp131 values less than ('STANDARD BRUSHED COPPER')
partition ptp132 values less than ('STANDARD BRUSHED NICKEL')
partition ptp133 values less than ('STANDARD BRUSHED STEEL')
partition ptp134 values less than ('STANDARD BRUSHED TIN')
partition ptp135 values less than ('STANDARD BURNISHED BRASS')
partition ptp136 values less than ('STANDARD BURNISHED COPPER')
partition ptp137 values less than ('STANDARD BURNISHED NICKEL')
partition ptp138 values less than ('STANDARD BURNISHED STEEL')
partition ptp139 values less than ('STANDARD BURNISHED TIN')
partition ptp140 values less than ('STANDARD PLATED BRASS')
partition ptp141 values less than ('STANDARD PLATED COPPER')
partition ptp142 values less than ('STANDARD PLATED NICKEL')
partition ptp143 values less than ('STANDARD PLATED STEEL')
partition ptp144 values less than ('STANDARD PLATED TIN')
partition ptp145 values less than ('STANDARD POLISHED BRASS')
partition ptp146 values less than ('STANDARD POLISHED COPPER')
partition ptp147 values less than ('STANDARD POLISHED NICKEL')
partition ptp148 values less than ('STANDARD POLISHED STEEL')
partition ptp149 values less than ('STANDARD POLISHED TIN')
partition ptp150 values less than (MAXVALUE)
)
}
}
*wait
*sql
{
connect tpcd/tpcd;
drop index p_cbp;
create index p_cbp
on parts (p_container,p_brand,p_partkey)
pctfree 1
nologging
compute statistics
tablespace ts_ind_pcbp
storage (initial 10m next 10m maxextents unlimited pctincrease 0)
parallel (degree 160)
global partition by range (p_container,p_brand)
(
partition pcbp1 values less than ('JUMBO BAG','Brand#111')
partition pcbp2 values less than ('JUMBO BAG','Brand#121')
partition pcbp3 values less than ('JUMBO BAG','Brand#131')
partition pcbp4 values less than ('JUMBO BAG','Brand#141')
partition pcbp5 values less than ('JUMBO BAG','Brand#151')
partition pcbp6 values less than ('JUMBO BAG','Brand#211')
partition pcbp7 values less than ('JUMBO BAG','Brand#221')
partition pcbp8 values less than ('JUMBO BAG','Brand#231')
partition pcbp9 values less than ('JUMBO BAG','Brand#241')
partition pcbp10 values less than ('JUMBO BAG','Brand#251')
partition pcbp11 values less than ('JUMBO BAG','Brand#311')
partition pcbp12 values less than ('JUMBO BAG','Brand#321')
partition pcbp13 values less than ('JUMBO BAG','Brand#331')
partition pcbp14 values less than ('JUMBO BAG','Brand#341')
partition pcbp15 values less than ('JUMBO BAG','Brand#351')
partition pcbp16 values less than ('JUMBO BAG','Brand#411')
partition pcbp17 values less than ('JUMBO BAG','Brand#421')
partition pcbp18 values less than ('JUMBO BAG','Brand#431')
partition pcbp19 values less than ('JUMBO BAG','Brand#441')
partition pcbp20 values less than ('JUMBO BAG','Brand#451')
partition pcbp21 values less than ('JUMBO BAG','Brand#511')
partition pcbp22 values less than ('JUMBO BAG','Brand#521')
partition pcbp23 values less than ('JUMBO BAG','Brand#531')
partition pcbp24 values less than ('JUMBO BAG','Brand#541')
partition pcbp25 values less than ('JUMBO BAG','Brand#551')
partition pcbp26 values less than ('JUMBO BOX','Brand#111')
partition pcbp27 values less than ('JUMBO BOX','Brand#121')
partition pcbp28 values less than ('JUMBO BOX','Brand#131')
partition pcbp29 values less than ('JUMBO BOX','Brand#141')
partition pcbp30 values less than ('JUMBO BOX','Brand#151')
partition pcbp31 values less than ('JUMBO BOX','Brand#211')
partition pcbp32 values less than ('JUMBO BOX','Brand#221')
partition pcbp33 values less than ('JUMBO BOX','Brand#231')
partition pcbp34 values less than ('JUMBO BOX','Brand#241')
partition pcbp35 values less than ('JUMBO BOX','Brand#251')
partition pcbp36 values less than ('JUMBO BOX','Brand#311')
partition pcbp37 values less than ('JUMBO BOX','Brand#321')
partition pcbp38 values less than ('JUMBO BOX','Brand#331')
partition pcbp39 values less than ('JUMBO BOX','Brand#341')
partition pcbp40 values less than ('JUMBO BOX','Brand#351')
partition pcbp41 values less than ('JUMBO BOX','Brand#411')
partition pcbp42 values less than ('JUMBO BOX','Brand#421')
partition pcbp43 values less than ('JUMBO BOX','Brand#431')
partition pcbp44 values less than ('JUMBO BOX','Brand#441')
partition pcbp45 values less than ('JUMBO BOX','Brand#451')
partition pcbp46 values less than ('JUMBO BOX','Brand#511')
partition pcbp47 values less than ('JUMBO BOX','Brand#521')
partition pcbp48 values less than ('JUMBO BOX','Brand#531')
partition pcbp49 values less than ('JUMBO BOX','Brand#541')
partition pcbp50 values less than ('JUMBO BOX','Brand#551')
partition pcbp51 values less than ('JUMBO CAN','Brand#111')
partition pcbp52 values less than ('JUMBO CAN','Brand#121')
partition pcbp53 values less than ('JUMBO CAN','Brand#131')
partition pcbp54 values less than ('JUMBO CAN','Brand#141')
partition pcbp55 values less than ('JUMBO CAN','Brand#151')
partition pcbp56 values less than ('JUMBO CAN','Brand#211')
partition pcbp57 values less than ('JUMBO CAN','Brand#221')
partition pcbp58 values less than ('JUMBO CAN','Brand#231')
partition pcbp59 values less than ('JUMBO CAN','Brand#241')
partition pcbp60 values less than ('JUMBO CAN','Brand#251')
partition pcbp61 values less than ('JUMBO CAN','Brand#311')
partition pcbp62 values less than ('JUMBO CAN','Brand#321')
partition pcbp63 values less than ('JUMBO CAN','Brand#331')
partition pcbp64 values less than ('JUMBO CAN','Brand#341')
partition pcbp65 values less than ('JUMBO CAN','Brand#351')

```


80

[illegible]

[illegible]

[illegible]

[illegible]

86

```

partition pcbp954 values less than ('WRAP PACK','Brand#141')
partition pcbp955 values less than ('WRAP PACK','Brand#151')
partition pcbp956 values less than ('WRAP PACK','Brand#211')
partition pcbp957 values less than ('WRAP PACK','Brand#221')
partition pcbp958 values less than ('WRAP PACK','Brand#231')
partition pcbp959 values less than ('WRAP PACK','Brand#241')
partition pcbp960 values less than ('WRAP PACK','Brand#251')
partition pcbp961 values less than ('WRAP PACK','Brand#311')
partition pcbp962 values less than ('WRAP PACK','Brand#321')
partition pcbp963 values less than ('WRAP PACK','Brand#331')
partition pcbp964 values less than ('WRAP PACK','Brand#341')
partition pcbp965 values less than ('WRAP PACK','Brand#351')
partition pcbp966 values less than ('WRAP PACK','Brand#411')
partition pcbp967 values less than ('WRAP PACK','Brand#421')
partition pcbp968 values less than ('WRAP PACK','Brand#431')
partition pcbp969 values less than ('WRAP PACK','Brand#441')
partition pcbp970 values less than ('WRAP PACK','Brand#451')
partition pcbp971 values less than ('WRAP PACK','Brand#511')
partition pcbp972 values less than ('WRAP PACK','Brand#521')
partition pcbp973 values less than ('WRAP PACK','Brand#531')
partition pcbp974 values less than ('WRAP PACK','Brand#541')
partition pcbp975 values less than ('WRAP PACK','Brand#551')
partition pcbp976 values less than ('WRAP PKG','Brand#111')
partition pcbp977 values less than ('WRAP PKG','Brand#121')
partition pcbp978 values less than ('WRAP PKG','Brand#131')
partition pcbp979 values less than ('WRAP PKG','Brand#141')
partition pcbp980 values less than ('WRAP PKG','Brand#151')
partition pcbp981 values less than ('WRAP PKG','Brand#211')
partition pcbp982 values less than ('WRAP PKG','Brand#221')
partition pcbp983 values less than ('WRAP PKG','Brand#231')
partition pcbp984 values less than ('WRAP PKG','Brand#241')
partition pcbp985 values less than ('WRAP PKG','Brand#251')
partition pcbp986 values less than ('WRAP PKG','Brand#311')
partition pcbp987 values less than ('WRAP PKG','Brand#321')
partition pcbp988 values less than ('WRAP PKG','Brand#331')
partition pcbp989 values less than ('WRAP PKG','Brand#341')
partition pcbp990 values less than ('WRAP PKG','Brand#351')
partition pcbp991 values less than ('WRAP PKG','Brand#411')
partition pcbp992 values less than ('WRAP PKG','Brand#421')
partition pcbp993 values less than ('WRAP PKG','Brand#431')
partition pcbp994 values less than ('WRAP PKG','Brand#441')
partition pcbp995 values less than ('WRAP PKG','Brand#451')
partition pcbp996 values less than ('WRAP PKG','Brand#511')
partition pcbp997 values less than ('WRAP PKG','Brand#521')
partition pcbp998 values less than ('WRAP PKG','Brand#531')
partition pcbp999 values less than ('WRAP PKG','Brand#541')
partition pcbp1000 values less than (MAXVALUE,MAXVALUE)
)
}
}
*wait
*sql
{
connect tpcd/tpcd;
drop index i_n_nationkey;
create unique index i_n_nationkey
on nation (n_nationkey)
pctfree 0
nologging
compute statistics
tablespace ts_constr
storage (initial 32k next 32k pctincrease 0)
}
}
*sql

{
connect tpcd/tpcd;
drop index i_r_regionkey;
create unique index i_r_regionkey
on region (r_regionkey)
pctfree 0
nologging
compute statistics
tablespace ts_constr
storage (initial 32k next 32k pctincrease 0)
}
}
*sql
{
connect tpcd/tpcd;
drop index i_s_suppkey;
create unique index i_s_suppkey
on supplier (s_suppkey)
pctfree 0
nologging
compute statistics
tablespace ts_constr
storage (initial 1m next 1m pctincrease 0)
parallel
}
}
*wait
*sql
{
connect tpcd/tpcd;
drop index i_c_custkey;
create unique index i_c_custkey
on customer (c_custkey)
pctfree 0
nologging
compute statistics
tablespace ts_constr
storage (initial 1m next 1m pctincrease 0)
parallel
}
}
*wait
*sql
{
connect tpcd/tpcd;
drop index i_ps_partkey_suppkey;
create unique index i_ps_partkey_suppkey
on partsupp (ps_partkey,ps_suppkey)
pctfree 0
nologging
compute statistics
tablespace ts_constr
storage (initial 1m next 1m pctincrease 0)
parallel
}
}
*wait
*sql
{
connect tpcd/tpcd;
drop index i_p_partkey;
create unique index i_p_partkey
on parts (p_partkey)
pctfree 0
nologging
compute statistics
tablespace ts_constr
storage (initial 1m next 1m pctincrease 0)
parallel
}
}
*wait
*sql
{
connect tpcd/tpcd;
alter table region drop constraint rp_k_r_regionkey;
alter table region add constraint rp_k_r_regionkey
primary key (r_regionkey) using index;
}
}
*sql
{
connect tpcd/tpcd;
alter table nation drop constraint np_k_n_nationkey;
alter table nation add constraint np_k_n_nationkey
primary key (n_nationkey) using index;
}
}
*sql
{
connect tpcd/tpcd;
alter table supplier drop constraint sp_k_s_suppkey;
alter table supplier add constraint sp_k_s_suppkey
primary key (s_suppkey) using index;
}
}
*sql
{
connect tpcd/tpcd;
alter table customer drop constraint cp_k_c_custkey;
alter table customer add constraint cp_k_c_custkey
primary key (c_custkey) using index;
}
}
*sql
{
connect tpcd/tpcd;
alter table parts drop constraint pp_k_p_partkey;
alter table parts add constraint pp_k_p_partkey
primary key (p_partkey) using index;
}
}
*sql
{
connect tpcd/tpcd;
alter table partsupp drop constraint psp_k_ps_partkey_suppkey;

```

```

alter table partsupp add constraint pspk_ps_partkey_supkey
    primary key (ps_partkey,ps_supkey) using index;
}
*sql
{
connect tpcd/tpcd;
alter table orders drop constraint ook_o_orderkey;
alter table orders add constraint ook_o_orderkey
    primary key (o_orderkey) using index;
}
*wait
*time
*bgoff
%e-ixcre

```

tera_sctso_1.dat

```

#####
# preprocessing-like directives

%b-preproc

*sql
\echo "{}" > script*getenv(BUMPX_CTR).sql
\svrmgrl <<1
\set echo on;
\set termout on;
\connect internal;
\select to_char(sysdate, 'MM-DD-YYYY HH24:MI:SS') now from dual;
\@script*getenv(BUMPX_CTR).sql;
\select to_char(sysdate, 'MM-DD-YYYY HH24:MI:SS') now from dual;
\exit;
\!
\bin/rm script*getenv(BUMPX_CTR).sql;

*load
\sqlldr {}

*mknod
\mknod {}

*dbgen
\setenv DSS_PATH {}
\ohome/100g/tpcd/dbgen/dbgen {}

*check_stop
\ohome/100g/checkstop.sh

*time
\echo
\echo {}
\date
\echo

*sh
\{}

%e-preproc
%b-sctso
*bgon=80
#####
# Schema Creation Phase - datafiles only (no tables or users)
# creating data tablespaces, datafiles
# creating tpcd's ts_default tablespace
*time
{
START BUILDING TABLESPACES
}
*sql
{
drop tablespace ts_default;
create tablespace ts_default
datafile '/ora81/oracle814/980901/dbs/ddisks/ts_default_1' size 5000m reuse
default storage (initial 32k next 1m maxextents unlimited pctincrease 0)
;
}
# creating tpcd's ts_l tablespace
*sql
{
drop tablespace ts_l;
create tablespace ts_l
datafile '/ora81/oracle814/980901/dbs/ddisks/ts_l_1' size 32000m reuse
default storage (initial 32k next 30m maxextents unlimited pctincrease 0)
;
}
# creating tpcd's ts_c tablespace
*sql
{
drop tablespace ts_c;
create tablespace ts_c
datafile '/ora81/oracle814/980901/dbs/ddisks/ts_c_1' size 6500m reuse
extent management local autoallocate
;
}
# creating tpcd's ts_o tablespace
*sql
{
drop tablespace ts_o;
create tablespace ts_o
datafile '/ora81/oracle814/980901/dbs/ddisks/ts_o_1' size 8400m reuse
default storage (initial 32k next 20m maxextents unlimited pctincrease 0)
;
}
# creating tpcd's ts_p tablespace
*sql
{
drop tablespace ts_p;
create tablespace ts_p

```

```

datafile '/ora81/oracle814/980901/dbs/ddisks/ts_p_1' size 6000m reuse
default storage (initial 32k next 10m maxextents unlimited pctincrease 0)
;
}
# creating tpcd's ts_ps tablespace
*sql
{
drop tablespace ts_ps;
create tablespace ts_ps
datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ps_1' size 30500m reuse
default storage (initial 32k next 62m pctincrease 0)
;
}
# creating tpcd's ts_s tablespace
*sql
{
drop tablespace ts_s;
create tablespace ts_s
datafile '/ora81/oracle814/980901/dbs/ddisks/ts_s_1' size 800m reuse
default storage (initial 32k next 10m pctincrease 0)
;
}
# creating tpcd's ts_ma_lineitem tablespace
*sql
{
drop tablespace ts_ma_lineitem;
create tablespace ts_ma_lineitem
datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ma_lineitem_1' size 800m reuse
default storage (initial 32k next 10m maxextents unlimited pctincrease 0)
;
}
# creating tpcd's ts_mj_locs tablespace
*sql
{
drop tablespace ts_mj_locs;
create tablespace ts_mj_locs
datafile '/ora81/oracle814/980901/dbs/ddisks/ts_mj_locs_1' size 31800m reuse
default storage (initial 32k next 45m maxextents unlimited pctincrease 0)
;
}
# creating tpcd's ts_mlog tablespace
*sql
{
drop tablespace ts_mlog;
create tablespace ts_mlog
datafile '/ora81/oracle814/980901/dbs/ddisks/ts_mlog_1' size 2000m reuse
default storage (initial 32k next 5m maxextents unlimited pctincrease 0)
;
}
# creating tpcd's ts_molog tablespace
*sql
{
drop tablespace ts_molog;
create tablespace ts_molog
datafile '/ora81/oracle814/980901/dbs/ddisks/ts_molog_1' size 500m reuse
default storage (initial 32k next 5m maxextents unlimited pctincrease 0)
;
}
# creating tpcd's ts_mjlog tablespace
*sql
{
drop tablespace ts_mjlog;
create tablespace ts_mjlog
datafile '/ora81/oracle814/980901/dbs/ddisks/ts_mjlog_1' size 2500m reuse
default storage (initial 32k next 5m maxextents unlimited pctincrease 0)
;
}
# creating tpcd's ts_mjlog tablespace
*sql
{
drop tablespace ts_mjlog;
create tablespace ts_mjlog
datafile '/ora81/oracle814/980901/dbs/ddisks/ts_mjlog_1' size 2500m reuse
default storage (initial 32k next 5m maxextents unlimited pctincrease 0)
;
}
# creating tpcd's ts_mj_ppssrn tablespace
*sql
{
drop tablespace ts_mj_ppssrn;
create tablespace ts_mj_ppssrn
datafile '/ora81/oracle814/980901/dbs/ddisks/ts_mj_ppssrn_1' size 23000m reuse
default storage (initial 32k next 20m maxextents unlimited pctincrease 0)
;
}
# creating tpcd's ts_mj_l_log tablespace
*sql
{
drop tablespace ts_mj_l_log;
create tablespace ts_mj_l_log
datafile '/ora81/oracle814/980901/dbs/ddisks/ts_mj_l_log_1' size 2000m reuse
default storage (initial 32k next 5m maxextents unlimited pctincrease 0)
;
}
# creating tpcd's ts_ij_l_rdate tablespace
*sql
{
drop tablespace ts_ij_l_pkey;
create tablespace ts_ij_l_pkey
datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_pkey_1' size 30000m reuse
default storage (initial 20m next 20m maxextents unlimited pctincrease 0)
;
}
# creating tpcd's ts_ij_l_sdate tablespace
*sql
{
drop tablespace ts_ij_l_sdate;
create tablespace ts_ij_l_sdate
datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_sdate_1' size 30000m reuse
default storage (initial 5m next 5m maxextents unlimited pctincrease 0)
;
}

```

```

}
# creating tpcd's ts_ij_l_ptp tablespace
*sql
{
drop tablespace ts_ij_l_ptp;
create tablespace ts_ij_l_ptp
datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_ptp_1' size 30000m reuse
default storage (initial 5m next 5m maxextents unlimited pctincrease 0)
;
}
*sql
{
drop tablespace ts_ij_l_rdate;
create tablespace ts_ij_l_rdate
datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_rdate_1' size 24970m reuse
default storage (initial 20m next 20m maxextents unlimited pctincrease 0)
}
# creating tpcd's ts_ind_lored tablespace
*sql
{
drop tablespace ts_ind_lored;
create tablespace ts_ind_lored
datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ind_lored_1' size 30000m reuse
default storage (initial 50m next 50m maxextents unlimited pctincrease 0)
;
}
# creating tpcd's ts_ij_l_odate tablespace
*sql
{
drop tablespace ts_ij_l_odate;
create tablespace ts_ij_l_odate
datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_odate_1' size 23550m reuse
default storage (initial 20m next 20m maxextents unlimited pctincrease 0)
;
}
# creating tpcd's ts_ij_l_lrid tablespace
*sql
{
drop tablespace ts_ij_l_lrid;
create tablespace ts_ij_l_lrid
datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_lrid_1' size 20800m reuse
default storage (initial 10m next 10m maxextents unlimited pctincrease 0)
;
}
# creating tpcd's ts_ij_l_orid tablespace
*sql
{
drop tablespace ts_ij_l_orid;
create tablespace ts_ij_l_orid
datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_orid_1' size 20850m reuse
default storage (initial 10m next 10m maxextents unlimited pctincrease 0)
;
}
# creating tpcd's ts_ij_l_nullrid tablespace
*sql
{
drop tablespace ts_ij_l_nullrid;
create tablespace ts_ij_l_nullrid
datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_nullrid_1' size 100m reuse
default storage (initial 10m next 10m maxextents unlimited pctincrease 0)
;
}
# creating tpcd's ts_constr tablespace
*sql
{
drop tablespace ts_constr;
create tablespace ts_constr
datafile '/ora81/oracle814/980901/dbs/ddisks/ts_constr_1' size 7000m reuse
default storage (initial 10m next 10m maxextents unlimited pctincrease 0)
;
}
# creating tpcd's ts_ij_p_ppsn tablespace
*sql
{
drop tablespace ts_ij_p_ppsn;
create tablespace ts_ij_p_ppsn
datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_p_ppsn_1' size 12300m reuse
default storage (initial 20m next 20m maxextents unlimited pctincrease 0)
;
}
# creating tpcd's ts_ij_p_nnpsps tablespace
*sql
{
drop tablespace ts_ij_p_nnpsps;
create tablespace ts_ij_p_nnpsps
datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_p_nnpsps_1' size 9880m reuse
default storage (initial 20m next 20m maxextents unlimited pctincrease 0)
;
}
*sql
{
drop tablespace ts_ij_p_nnps;
create tablespace ts_ij_p_nnps
datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_odate_25' size 23550m reuse
default storage (initial 20m next 20m maxextents unlimited pctincrease 0)
;
}
# creating tpcd's ts_ind_oclokod tablespace
*sql
{
drop tablespace ts_ind_oclokod;
create tablespace ts_ind_oclokod
datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ind_oclokod_1' size 19600m reuse
default storage (initial 10m next 10m maxextents unlimited pctincrease 0)
;
}
# creating tpcd's ts_ind_ookey tablespace
*sql
{
drop tablespace ts_ind_ookey;
create tablespace ts_ind_ookey
datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ind_ookey_1' size 9250m reuse
default storage (initial 100m next 100m maxextents unlimited pctincrease 0)
;
}
# creating tpcd's ts_ind_pcbp tablespace
*sql
{
drop tablespace ts_ind_pcbp;
create tablespace ts_ind_pcbp
datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ind_pcbp_1' size 3000m reuse
default storage (initial 1m next 4m maxextents unlimited pctincrease 0)
;
}
# creating tpcd's ts_temp tablespace
*sql
{
drop tablespace ts_temp;
create temporary tablespace ts_temp
tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_temp_1' size 30000m reuse
extent management local
uniform size 200m
;
}
*wait
# adding tpcd's ts_default datafiles
*sql
{
alter tablespace ts_default
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_default_2' size 5000m reuse;
}
*sql
{
alter tablespace ts_default
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_default_3' size 5000m reuse;
}
*sql
{
alter tablespace ts_default
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_default_4' size 5000m reuse;
}
# adding tpcd's ts_l datafiles
*sql
{
alter tablespace ts_l
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_l_2' size 32000m reuse;
}
*sql
{
alter tablespace ts_l
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_l_3' size 32000m reuse;
}
*sql
{
alter tablespace ts_l
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_l_4' size 32000m reuse;
}
*sql
{
alter tablespace ts_l
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_l_5' size 32000m reuse;
}
*sql
{
alter tablespace ts_l
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_l_6' size 32000m reuse;
}
*sql
{
alter tablespace ts_l
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_l_7' size 32000m reuse;
}
*sql
{
alter tablespace ts_l
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_l_8' size 32000m reuse;
}
*sql
{
alter tablespace ts_l
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_l_9' size 32000m reuse;
}
*sql
{
alter tablespace ts_l
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_l_10' size 32000m reuse;
}
*sql
{
alter tablespace ts_l
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_l_11' size 32000m reuse;
}
*sql
{
alter tablespace ts_l
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_l_12' size 32000m reuse;
}
*sql
{
alter tablespace ts_l
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_l_13' size 32000m reuse;
}
*sql
{
alter tablespace ts_l
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_l_14' size 32000m reuse;
}
*sql
{
alter tablespace ts_l
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_l_15' size 32000m reuse;
}
}

```



```

}
*sql
{
alter tablespace ts_temp
add tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_temp_22' size 30000m reuse;
}
*sql
{
alter tablespace ts_temp
add tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_temp_23' size 30000m reuse;
}
*sql
{
alter tablespace ts_temp
add tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_temp_24' size 30000m reuse;
}
*sql
{
alter tablespace ts_temp
add tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_temp_25' size 30000m reuse;
}
*sql
{
alter tablespace ts_temp
add tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_temp_26' size 30000m reuse;
}
*sql
{
alter tablespace ts_temp
add tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_temp_27' size 30000m reuse;
}
*sql
{
alter tablespace ts_temp
add tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_temp_28' size 30000m reuse;
}
*sql
{
alter tablespace ts_temp
add tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_temp_29' size 30000m reuse;
}
*sql
{
alter tablespace ts_temp
add tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_temp_30' size 30000m reuse;
}
*sql
{
alter tablespace ts_temp
add tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_temp_31' size 30000m reuse;
}
*sql
{
alter tablespace ts_temp
add tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_temp_32' size 30000m reuse;
}
*wait
*time
{
DONE BUILDING TABLESPACES
}
*wait
*bgoff
%e-sctso

```

tera_sctso_2.dat

```

#####
# preprocessing-like directives

%b-preproc

*sql
\echo "{}" > script*getenv(BUMPX_CTR).sql
\svrmgrl <<!
\set echo on;
\set termout on;
\connect internal;
\select to_char(sysdate, 'MM-DD-YYYY HH24:MI:SS') now from dual;
@script*getenv(BUMPX_CTR).sql;
\select to_char(sysdate, 'MM-DD-YYYY HH24:MI:SS') now from dual;
\exit;
\!
\bin/rm script*getenv(BUMPX_CTR).sql;

*load
\sqlldr {}

*mknod
\mknod {}

*dbgen
\setenv DSS_PATH {}
\ohome/100g/tpcd/dbgen/dbgen {}

*check_stop
\ohome/100g/checkstop.sh

*time
\echo
\echo {}
\date
\echo

*sh
\{}

%e-preproc

```

```

%b-sctso
*bgon=80
#####
# Schema Creation Phase - datafiles only (no tables or users)
# creating data tablespaces, datafiles
# creating tpcd's ts_ij_l_pkey tablespace
*time
{
START ADDING TABLESPACES
}
*sql
{
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_pkey_2' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_pkey_3' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_pkey_4' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_pkey_5' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_pkey_6' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_pkey_7' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_pkey_8' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_pkey_9' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_pkey_10' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_pkey_11' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_pkey_12' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_pkey_13' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_pkey_14' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_pkey_15' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_pkey_16' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_pkey_17' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_pkey_18' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_pkey_19' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_pkey_20' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_sdate_2' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_sdate_3' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_sdate_4' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_sdate_5' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_sdate_6' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_sdate_7' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_sdate_8' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_sdate_9' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_sdate_10' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_sdate_11' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_sdate_12' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_sdate_13' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_sdate_14' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_sdate_15' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_sdate_16' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_sdate_17' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_sdate_18' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_sdate_19' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_sdate_20' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ind_lored_2' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ind_lored_3' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ind_lored_4' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ind_lored_5' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ind_lored_6' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ind_lored_7' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ind_lored_8' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_rdate_2' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_rdate_3' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_rdate_4' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_rdate_5' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_rdate_6' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_rdate_7' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_rdate_8' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_rdate_9' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_rdate_10' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_rdate_11' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_rdate_12' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_rdate_13' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_rdate_14' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_rdate_15' drop;
alter database tempfile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_rdate_16' drop;
}
*wait
*time
# adding tpcd's ts_ij_l_pkey datafiles
*sql
{
alter tablespace ts_ij_l_pkey
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_pkey_2' size 30000m reuse;
}
*sql
{
alter tablespace ts_ij_l_pkey
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_pkey_3' size 30000m reuse;
}
*sql
{
alter tablespace ts_ij_l_pkey
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_pkey_4' size 30000m reuse;
}
*sql
{
alter tablespace ts_ij_l_pkey
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_pkey_5' size 30000m reuse;
}
}

```



```

add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ind_lored_2' size 30000m
reuse;
}
*sql
{
alter tablespace ts_ind_lored
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ind_lored_3' size 30000m
reuse;
}
*sql
{
alter tablespace ts_ind_lored
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ind_lored_4' size 30000m
reuse;
}
*sql
{
alter tablespace ts_ind_lored
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ind_lored_5' size 30000m
reuse;
}
*sql
{
alter tablespace ts_ind_lored
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ind_lored_6' size 30000m
reuse;
}
*sql
{
alter tablespace ts_ind_lored
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ind_lored_7' size 30000m
reuse;
}
*sql
{
alter tablespace ts_ind_lored
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ind_lored_8' size 30000m
reuse;
}
# adding tpcd's ts_ij_l_rdate datafiles
*sql
{
alter tablespace ts_ij_l_rdate
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_rdate_2' size 24970m
reuse;
}
*sql
{
alter tablespace ts_ij_l_rdate
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_rdate_3' size 24970m
reuse;
}
*sql
{
alter tablespace ts_ij_l_rdate
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_rdate_4' size 24970m
reuse;
}
*sql
{
alter tablespace ts_ij_l_rdate
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_rdate_5' size 24970m
reuse;
}
*sql
{
alter tablespace ts_ij_l_rdate
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_rdate_6' size 24970m
reuse;
}
*sql
{
alter tablespace ts_ij_l_rdate
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_rdate_7' size 24970m
reuse;
}
*sql
{
alter tablespace ts_ij_l_rdate
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_rdate_8' size 24970m
reuse;
}
*sql
{
alter tablespace ts_ij_l_rdate
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_rdate_9' size 24970m
reuse;
}
*sql
{
alter tablespace ts_ij_l_rdate
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_rdate_10' size 24970m
reuse;
}
*sql
{
alter tablespace ts_ij_l_rdate
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_rdate_11' size 24970m
reuse;
}
*sql
{
alter tablespace ts_ij_l_rdate
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_rdate_12' size 24970m
reuse;
}
*sql
{
alter tablespace ts_ij_l_rdate
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_rdate_13' size 24970m
reuse;
}
*sql
{
alter tablespace ts_ij_l_rdate
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_rdate_14' size 24970m
reuse;
}
}

```

```

*sql
{
alter tablespace ts_ij_l_rdate
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_rdate_15' size 24970m
reuse;
}
*sql
{
alter tablespace ts_ij_l_rdate
add datafile '/ora81/oracle814/980901/dbs/ddisks/ts_ij_l_rdate_16' size 24970m
reuse;
}
*wait
*time
{
DONE ADDING TABLESPACES
}
*bgoff
%e-sctso

```

tera_scuto.dat

```

#####
# preprocessing-like directives

%b-preproc

*sql
\echo "*" > script*getenv(BUMPX_CTR).sql
\svrmgrl <<!
\set echo on;
\set termout on;
\connect internal;
\select to_char(sysdate, 'MM-DD-YYYY HH24:MI:SS') now from dual;
\@script*getenv(BUMPX_CTR).sql;
\select to_char(sysdate, 'MM-DD-YYYY HH24:MI:SS') now from dual;
\exit;
\!
\bin/rm script*getenv(BUMPX_CTR).sql;

*load
\sqlldr {}

*mknod
\mknod {}

*dbgen
\setenv DSS_PATH {}
\ohome/100g/tpcd/dbgen/dbgen {}

*check_stop
\ohome/100g/checkstop.sh

*time
\echo
\echo {}
\date
\echo

*sh
\{}

%e-preproc
%b-scuto
*bgon=1
#####
# Schema Creation Phase - User and Tables ONLY (no datafiles)
# creating tpcd user
*sql
{
shutdown
}
*sql
{
startup pfile=/ora81/oracle814/980901/dbs/db_build.ora;
}
*time
{
START CREATING SCHEMA. START OF LOAD TIME
}
*sql
{
drop user tpcd cascade;
grant DBA
to tpcd identified by tpcd;
}
*wait
*sql
{
connect tpcd/tpcd;
}
*wait
*sql
{
connect tpcd/tpcd;
drop table lineitem;
create table lineitem(
l_shipdate          date ,
l_orderkey          number NOT NULL,
l_discount          number ,
l_extendedprice     number ,
l_suppkey           number NOT NULL,
l_quantity          number ,
l_returnflag        char(1) ,
l_partkey           number NOT NULL,
l_linestatus        char(1) ,
l_tax              number NOT NULL,
l_commitdate        date ,
l_receiptdate       date ,

```



```

,
partition item105 values less than (to_date('1996-07-16','YYYY-MM-DD'))
,
partition item106 values less than (to_date('1996-08-01','YYYY-MM-DD'))
,
partition item107 values less than (to_date('1996-08-16','YYYY-MM-DD'))
,
partition item108 values less than (to_date('1996-09-01','YYYY-MM-DD'))
,
partition item109 values less than (to_date('1996-09-16','YYYY-MM-DD'))
,
partition item110 values less than (to_date('1996-10-01','YYYY-MM-DD'))
,
partition item111 values less than (to_date('1996-10-16','YYYY-MM-DD'))
,
partition item112 values less than (to_date('1996-11-01','YYYY-MM-DD'))
,
partition item113 values less than (to_date('1996-11-16','YYYY-MM-DD'))
,
partition item114 values less than (to_date('1996-12-01','YYYY-MM-DD'))
,
partition item115 values less than (to_date('1996-12-16','YYYY-MM-DD'))
,
partition item116 values less than (to_date('1997-01-01','YYYY-MM-DD'))
,
partition item117 values less than (to_date('1997-01-16','YYYY-MM-DD'))
,
partition item118 values less than (to_date('1997-02-01','YYYY-MM-DD'))
,
partition item119 values less than (to_date('1997-02-16','YYYY-MM-DD'))
,
partition item120 values less than (to_date('1997-03-01','YYYY-MM-DD'))
,
partition item121 values less than (to_date('1997-03-16','YYYY-MM-DD'))
,
partition item122 values less than (to_date('1997-04-01','YYYY-MM-DD'))
,
partition item123 values less than (to_date('1997-04-16','YYYY-MM-DD'))
,
partition item124 values less than (to_date('1997-05-01','YYYY-MM-DD'))
,
partition item125 values less than (to_date('1997-05-16','YYYY-MM-DD'))
,
partition item126 values less than (to_date('1997-06-01','YYYY-MM-DD'))
,
partition item127 values less than (to_date('1997-06-16','YYYY-MM-DD'))
,
partition item128 values less than (to_date('1997-07-01','YYYY-MM-DD'))
,
partition item129 values less than (to_date('1997-07-16','YYYY-MM-DD'))
,
partition item130 values less than (to_date('1997-08-01','YYYY-MM-DD'))
,
partition item131 values less than (to_date('1997-08-16','YYYY-MM-DD'))
,
partition item132 values less than (to_date('1997-09-01','YYYY-MM-DD'))
,
partition item133 values less than (to_date('1997-09-16','YYYY-MM-DD'))
,
partition item134 values less than (to_date('1997-10-01','YYYY-MM-DD'))
,
partition item135 values less than (to_date('1997-10-16','YYYY-MM-DD'))
,
partition item136 values less than (to_date('1997-11-01','YYYY-MM-DD'))
,
partition item137 values less than (to_date('1997-11-16','YYYY-MM-DD'))
,
partition item138 values less than (to_date('1997-12-01','YYYY-MM-DD'))
,
partition item139 values less than (to_date('1997-12-16','YYYY-MM-DD'))
,
partition item140 values less than (to_date('1998-01-01','YYYY-MM-DD'))
,
partition item141 values less than (to_date('1998-01-16','YYYY-MM-DD'))
,
partition item142 values less than (to_date('1998-02-01','YYYY-MM-DD'))
,
partition item143 values less than (to_date('1998-02-16','YYYY-MM-DD'))
,
partition item144 values less than (to_date('1998-03-01','YYYY-MM-DD'))
,
partition item145 values less than (to_date('1998-03-16','YYYY-MM-DD'))
,
partition item146 values less than (to_date('1998-04-01','YYYY-MM-DD'))
,
partition item147 values less than (to_date('1998-04-16','YYYY-MM-DD'))
,
partition item148 values less than (to_date('1998-05-01','YYYY-MM-DD'))
,
partition item149 values less than (to_date('1998-05-16','YYYY-MM-DD'))
,
partition item150 values less than (to_date('1998-06-01','YYYY-MM-DD'))
,
partition item151 values less than (to_date('1998-06-16','YYYY-MM-DD'))
,
partition item152 values less than (to_date('1998-07-01','YYYY-MM-DD'))
,
partition item153 values less than (to_date('1998-07-16','YYYY-MM-DD'))
,
partition item154 values less than (to_date('1998-08-01','YYYY-MM-DD'))
,
partition item155 values less than (to_date('1998-08-16','YYYY-MM-DD'))
,
partition item156 values less than (to_date('1998-09-01','YYYY-MM-DD'))
,
partition item157 values less than (to_date('1998-09-16','YYYY-MM-DD'))
,
partition item158 values less than (to_date('1998-10-01','YYYY-MM-DD'))
,
partition item159 values less than (to_date('1998-11-01','YYYY-MM-DD'))
,

```

```

partition item160 values less than (MAXVALUE)
)
;
drop table orders;
create table orders(
    o_orderdate        date ,
    o_orderkey          number NOT NULL,
    o_custkey           number NOT NULL,
    o_orderpriority     varchar(15) ,
    o_shippriority     number ,
    o_clerk             varchar(15) ,
    o_orderstatus       char(1) ,
    o_totalprice        number ,
    o_comment           varchar(79)
)
pctfree 1
pctused 99
initrans 10
tablespace ts_o
storage (initial 32k next 10m maxextents unlimited freelists 32 freelist groups 5
pctincrease 0)
parallel (degree 160)
nologging
partition by hash (o_orderkey)
partitions 160
;
drop table partsupp;
create table partsupp(
    ps_partkey          number NOT NULL,
    ps_suppkey          number NOT NULL,
    ps_supplycost       number NOT NULL,
    ps_availqty         number ,
    ps_comment          varchar(199)
)
pctfree 0
pctused 99
tablespace ts_ps
storage (initial 32k next 48m maxextents unlimited pctincrease 0)
parallel
nologging
;
drop table parts;
create table parts(
    p_partkey           number NOT NULL,
    p_type              varchar(25) ,
    p_size              number ,
    p_brand              varchar(10) ,
    p_name              varchar(55) ,
    p_container         varchar(10) ,
    p_mfg               varchar(25) ,
    p_retailprice       number ,
    p_comment           varchar(23)
)
pctfree 0
pctused 99
tablespace ts_p
storage (initial 32k next 25m maxextents unlimited pctincrease 0)
parallel
nologging
partition by range (p_size)
(
    partition part1 values less than (2)
    ,
    partition part2 values less than (3)
    ,
    partition part3 values less than (4)
    ,
    partition part4 values less than (5)
    ,
    partition part5 values less than (6)
    ,
    partition part6 values less than (7)
    ,
    partition part7 values less than (8)
    ,
    partition part8 values less than (9)
    ,
    partition part9 values less than (10)
    ,
    partition part10 values less than (11)
    ,
    partition part11 values less than (12)
    ,
    partition part12 values less than (13)
    ,
    partition part13 values less than (14)
    ,
    partition part14 values less than (15)
    ,
    partition part15 values less than (16)
    ,
    partition part16 values less than (17)
    ,
    partition part17 values less than (18)
    ,
    partition part18 values less than (19)
    ,
    partition part19 values less than (20)
    ,
    partition part20 values less than (21)
    ,
    partition part21 values less than (22)
    ,
    partition part22 values less than (23)
    ,
    partition part23 values less than (24)
    ,
    partition part24 values less than (25)
    ,
    partition part25 values less than (26)
)
;

```

```

partition part26 values less than (27)
,
partition part27 values less than (28)
,
partition part28 values less than (29)
,
partition part29 values less than (30)
,
partition part30 values less than (31)
,
partition part31 values less than (32)
,
partition part32 values less than (33)
,
partition part33 values less than (34)
,
partition part34 values less than (35)
,
partition part35 values less than (36)
,
partition part36 values less than (37)
,
partition part37 values less than (38)
,
partition part38 values less than (39)
,
partition part39 values less than (40)
,
partition part40 values less than (41)
,
partition part41 values less than (42)
,
partition part42 values less than (43)
,
partition part43 values less than (44)
,
partition part44 values less than (45)
,
partition part45 values less than (46)
,
partition part46 values less than (47)
,
partition part47 values less than (48)
,
partition part48 values less than (49)
,
partition part49 values less than (50)
,
partition part50 values less than (MAXVALUE)
)
;
drop table customer;
create table customer(
    c_custkey          number NOT NULL,
    c_mktsegment       varchar(10) ,
    c_nationkey         number ,
    c_name              varchar(25) ,
    c_address           varchar(40) ,
    c_phone             varchar(15) ,
    c_acctbal           number ,
    c_comment           varchar(117)
)
;
pctfree 0
pctused 99
tablespace ts_c
parallel
nologging
partition by hash (c_custkey)
partitions 160
;
drop table supplier;
create table supplier(
    s_suppkey          number NOT NULL,
    s_nationkey         number ,
    s_comment           varchar(101) ,
    s_name              varchar(25) ,
    s_address           varchar(40) ,
    s_phone             varchar(15) ,
    s_acctbal           number
)
;
pctfree 0
pctused 99
tablespace ts_s
storage (initial 32k next 2m maxextents unlimited pctincrease 0)
parallel
nologging
;
drop table nation;
create table nation(
    n_nationkey         number NOT NULL,
    n_name              varchar(25) ,
    n_regionkey         number ,
    n_comment           varchar(152)
)
;
tablespace system
;
drop table region;
create table region(
    r_regionkey         number ,
    r_name              varchar(25) ,
    r_comment           varchar(152)
)
;
tablespace system
;
;
*wait
# altering tpcd's temp tablespace
*sql
{
alter user tpcd temporary tablespace ts_temp;
}

```

```

*wait
# altering tpcd's default tablespace
*sql
{
alter user tpcd default tablespace ts_default;
grant query rewrite to tpcd;
grant global query rewrite to tpcd;
}
*wait
*time
{
DONE CREATING SCHEMA
}
*wait
*bgoff
%e-scuto

```

tera_scuvo.dat

```

#####
# preprocessing-like directives

%b-preproc

*sql
\echo "{}" > script*getenv(BUMPX_CTR).sql
\svrmgrl <<!
\set echo on;
\set termout on;
\connect internal;
\select to_char(sysdate, 'MM-DD-YYYY HH24:MI:SS') now from dual;
\@script*getenv(BUMPX_CTR).sql;
\select to_char(sysdate, 'MM-DD-YYYY HH24:MI:SS') now from dual;
\exit;
\!
\bin/rm script*getenv(BUMPX_CTR).sql;

*load
\sqlldr {}

*mknod
\mknod {}

*dbgen
\setenv DSS_PATH {}
\ohome/100g/tpcd/dbgen/dbgen {}

*check_stop
\ohome/100g/checkstop.sh

*time
\echo
\echo {}
\date
\echo

*sh
\{}

%e-preproc
%b-scuvo
*bgon=1
#####
# Schema Creation Phase - Views ONLY (no datafiles)
#####
# First I will create the materialized log for the base tables
*time
{
START BUILDING VIEWS
}
*sql
{
shutdown;
}
*wait
*sql
{
startup pfile=/ora81/oracle814/980901/dbs/mv_build.ora;
}
*wait
*time
*sql
{
connect tpcd/tpcd;
drop materialized view log on lineitem;
create materialized view log on lineitem
tablespace ts_mlog
storage (initial 32k next 5m pctincrease 0)
partition by range (l_shipdate)
(
partition item1 values less than (to_date('1992-03-01','YYYY-MM-DD'))
,
partition item2 values less than (to_date('1992-03-24','YYYY-MM-DD'))
,
partition item3 values less than (to_date('1992-04-14','YYYY-MM-DD'))
,
partition item4 values less than (to_date('1992-05-01','YYYY-MM-DD'))
,
partition item5 values less than (to_date('1992-05-16','YYYY-MM-DD'))
,
partition item6 values less than (to_date('1992-06-01','YYYY-MM-DD'))
,
partition item7 values less than (to_date('1992-06-16','YYYY-MM-DD'))
,
partition item8 values less than (to_date('1992-07-01','YYYY-MM-DD'))
,
partition item9 values less than (to_date('1992-07-16','YYYY-MM-DD'))
,

```



```

partition item121 values less than (to_date('1997-03-16','YYYY-MM-DD'))
,
partition item122 values less than (to_date('1997-04-01','YYYY-MM-DD'))
,
partition item123 values less than (to_date('1997-04-16','YYYY-MM-DD'))
,
partition item124 values less than (to_date('1997-05-01','YYYY-MM-DD'))
,
partition item125 values less than (to_date('1997-05-16','YYYY-MM-DD'))
,
partition item126 values less than (to_date('1997-06-01','YYYY-MM-DD'))
,
partition item127 values less than (to_date('1997-06-16','YYYY-MM-DD'))
,
partition item128 values less than (to_date('1997-07-01','YYYY-MM-DD'))
,
partition item129 values less than (to_date('1997-07-16','YYYY-MM-DD'))
,
partition item130 values less than (to_date('1997-08-01','YYYY-MM-DD'))
,
partition item131 values less than (to_date('1997-08-16','YYYY-MM-DD'))
,
partition item132 values less than (to_date('1997-09-01','YYYY-MM-DD'))
,
partition item133 values less than (to_date('1997-09-16','YYYY-MM-DD'))
,
partition item134 values less than (to_date('1997-10-01','YYYY-MM-DD'))
,
partition item135 values less than (to_date('1997-10-16','YYYY-MM-DD'))
,
partition item136 values less than (to_date('1997-11-01','YYYY-MM-DD'))
,
partition item137 values less than (to_date('1997-11-16','YYYY-MM-DD'))
,
partition item138 values less than (to_date('1997-12-01','YYYY-MM-DD'))
,
partition item139 values less than (to_date('1997-12-16','YYYY-MM-DD'))
,
partition item140 values less than (to_date('1998-01-01','YYYY-MM-DD'))
,
partition item141 values less than (to_date('1998-01-16','YYYY-MM-DD'))
,
partition item142 values less than (to_date('1998-02-01','YYYY-MM-DD'))
,
partition item143 values less than (to_date('1998-02-16','YYYY-MM-DD'))
,
partition item144 values less than (to_date('1998-03-01','YYYY-MM-DD'))
,
partition item145 values less than (to_date('1998-03-16','YYYY-MM-DD'))
,
partition item146 values less than (to_date('1998-04-01','YYYY-MM-DD'))
,
partition item147 values less than (to_date('1998-04-16','YYYY-MM-DD'))
,
partition item148 values less than (to_date('1998-05-01','YYYY-MM-DD'))
,
partition item149 values less than (to_date('1998-05-16','YYYY-MM-DD'))
,
partition item150 values less than (to_date('1998-06-01','YYYY-MM-DD'))
,
partition item151 values less than (to_date('1998-06-16','YYYY-MM-DD'))
,
partition item152 values less than (to_date('1998-07-01','YYYY-MM-DD'))
,
partition item153 values less than (to_date('1998-07-16','YYYY-MM-DD'))
,
partition item154 values less than (to_date('1998-08-01','YYYY-MM-DD'))
,
partition item155 values less than (to_date('1998-08-16','YYYY-MM-DD'))
,
partition item156 values less than (to_date('1998-09-01','YYYY-MM-DD'))
,
partition item157 values less than (to_date('1998-09-16','YYYY-MM-DD'))
,
partition item158 values less than (to_date('1998-10-01','YYYY-MM-DD'))
,
partition item159 values less than (to_date('1998-11-01','YYYY-MM-DD'))
,
partition item160 values less than (MAXVALUE)
)
with rowid (
    l_shipdate
    , l_returnflag
    , l_linestatus
    , l_quantity
    , l_extendedprice
    , l_discount
    , l_tax
    , l_suppkey
) including new values
;
drop materialized view log on orders;
create materialized view log on orders
tablespace ts_molog
storage (initial 32k next 3m pctincrease 0)
partition by hash (o_orderkey)
partitions 160
with rowid (
    o_orderkey
) including new values
;
drop materialized view log on customer;
create materialized view log on customer
tablespace ts_maint
with rowid including new values ;
drop materialized view log on supplier;
create materialized view log on supplier
tablespace ts_maint
with rowid including new values ;
drop materialized view log on parts;
create materialized view log on parts

```

```

tablespace ts_maint
with rowid including new values ;
drop materialized view log on partsupp;
create materialized view log on partsupp
tablespace ts_maint
with rowid including new values ;
drop materialized view log on nation;
create materialized view log on nation
tablespace ts_maint
with rowid including new values ;
drop materialized view log on region;
create materialized view log on region
tablespace ts_maint
with rowid including new values ;
}
*wait
#####
# Now I can create the views with the corresponding materialized logs
*sql
{
connect tpcd/tpcd;
drop materialized view ma_lineitem;
create materialized view ma_lineitem
pctfree 2
tablespace ts_ma_lineitem
storage (initial 512k next 512k maxextents unlimited pctincrease 0)
nologging
BUILD immediate
REFRESH fast
ON commit
enable query rewrite
as select
    l_shipdate,
    l_returnflag,
    l_linestatus,
    sum(l_quantity) as sum_qty,
    count(l_quantity) as count_qty,
    sum(l_extendedprice) as sum_base_price,
    count(l_extendedprice) as count_price,
    sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
    count(l_extendedprice * (1 - l_discount)) as count_disc_price,
    sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge,
    count(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as count_charge,
    sum(l_discount) as sum_disc,
    count(l_discount) as count_disc,
    count(*) as count_order
FROM
    lineitem
GROUP BY
    l_returnflag,
    l_linestatus,
    l_shipdate
;
}
*sql
{
connect tpcd/tpcd;
drop materialized view ma_l_sdfy;
create materialized view ma_l_sdfy
pctfree 2
storage (initial 128k next 128k pctincrease 0)
nologging
BUILD immediate
REFRESH fast
ON commit
enable query rewrite
as
select
    l_discount,
    l_quantity,
    dfy(l_shipdate) as dfy,
    sum(l_extendedprice * l_discount) as volume,
    count(l_extendedprice * l_discount) as count_volume,
    count(*) as count_grp
from
    lineitem
group by
    l_discount,
    l_quantity,
    dfy(l_shipdate)
;
}
*sql
{
connect tpcd/tpcd;
drop materialized view mj_plocs;
create materialized view mj_plocs
pctfree 2
tablespace ts_mj_locs
storage (initial 1m next 200m maxextents unlimited pctincrease 0)
parallel(degree 161 )
nologging
partition by range (l_shipdate)
subpartition by hash(c_mktsegment)
subpartitions 5
(
    partition mj_locs1 values less than (to_date('1992-03-01','YYYY-MM-DD'))
    ,
    partition mj_locs2 values less than (to_date('1992-03-24','YYYY-MM-DD'))
    ,
    partition mj_locs3 values less than (to_date('1992-04-14','YYYY-MM-DD'))
    ,
    partition mj_locs4 values less than (to_date('1992-05-01','YYYY-MM-DD'))
    ,
    partition mj_locs5 values less than (to_date('1992-05-16','YYYY-MM-DD'))
    ,
    partition mj_locs6 values less than (to_date('1992-06-01','YYYY-MM-DD'))
    ,
    partition mj_locs7 values less than (to_date('1992-06-16','YYYY-MM-DD'))
    ,
    partition mj_locs8 values less than (to_date('1992-07-01','YYYY-MM-DD'))
)

```



```

partition mj_locs120 values less than (to_date('1997-03-01','YYYY-MM-DD'))
partition mj_locs121 values less than (to_date('1997-03-16','YYYY-MM-DD'))
partition mj_locs122 values less than (to_date('1997-04-01','YYYY-MM-DD'))
partition mj_locs123 values less than (to_date('1997-04-16','YYYY-MM-DD'))
partition mj_locs124 values less than (to_date('1997-05-01','YYYY-MM-DD'))
partition mj_locs125 values less than (to_date('1997-05-16','YYYY-MM-DD'))
partition mj_locs126 values less than (to_date('1997-06-01','YYYY-MM-DD'))
partition mj_locs127 values less than (to_date('1997-06-16','YYYY-MM-DD'))
partition mj_locs128 values less than (to_date('1997-07-01','YYYY-MM-DD'))
partition mj_locs129 values less than (to_date('1997-07-16','YYYY-MM-DD'))
partition mj_locs130 values less than (to_date('1997-08-01','YYYY-MM-DD'))
partition mj_locs131 values less than (to_date('1997-08-16','YYYY-MM-DD'))
partition mj_locs132 values less than (to_date('1997-09-01','YYYY-MM-DD'))
partition mj_locs133 values less than (to_date('1997-09-16','YYYY-MM-DD'))
partition mj_locs134 values less than (to_date('1997-10-01','YYYY-MM-DD'))
partition mj_locs135 values less than (to_date('1997-10-16','YYYY-MM-DD'))
partition mj_locs136 values less than (to_date('1997-11-01','YYYY-MM-DD'))
partition mj_locs137 values less than (to_date('1997-11-16','YYYY-MM-DD'))
partition mj_locs138 values less than (to_date('1997-12-01','YYYY-MM-DD'))
partition mj_locs139 values less than (to_date('1997-12-16','YYYY-MM-DD'))
partition mj_locs140 values less than (to_date('1998-01-01','YYYY-MM-DD'))
partition mj_locs141 values less than (to_date('1998-01-16','YYYY-MM-DD'))
partition mj_locs142 values less than (to_date('1998-02-01','YYYY-MM-DD'))
partition mj_locs143 values less than (to_date('1998-02-16','YYYY-MM-DD'))
partition mj_locs144 values less than (to_date('1998-03-01','YYYY-MM-DD'))
partition mj_locs145 values less than (to_date('1998-03-16','YYYY-MM-DD'))
partition mj_locs146 values less than (to_date('1998-04-01','YYYY-MM-DD'))
partition mj_locs147 values less than (to_date('1998-04-16','YYYY-MM-DD'))
partition mj_locs148 values less than (to_date('1998-05-01','YYYY-MM-DD'))
partition mj_locs149 values less than (to_date('1998-05-16','YYYY-MM-DD'))
partition mj_locs150 values less than (to_date('1998-06-01','YYYY-MM-DD'))
partition mj_locs151 values less than (to_date('1998-06-16','YYYY-MM-DD'))
partition mj_locs152 values less than (to_date('1998-07-01','YYYY-MM-DD'))
partition mj_locs153 values less than (to_date('1998-07-16','YYYY-MM-DD'))
partition mj_locs154 values less than (to_date('1998-08-01','YYYY-MM-DD'))
partition mj_locs155 values less than (to_date('1998-08-16','YYYY-MM-DD'))
partition mj_locs156 values less than (to_date('1998-09-01','YYYY-MM-DD'))
partition mj_locs157 values less than (to_date('1998-09-16','YYYY-MM-DD'))
partition mj_locs158 values less than (to_date('1998-10-01','YYYY-MM-DD'))
partition mj_locs159 values less than (to_date('1998-11-01','YYYY-MM-DD'))
partition mj_locs160 values less than (MAXVALUE)
)
enable row movement
BUILD immediate
REFRESH fast
ON commit
enable query rewrite
as select /*+ ordered */
    l_shipdate,
    o_orderdate,
    l_extendedprice,
    l_discount,
    c_custkey,
    s_suppkey,
    c_nationkey,
    s_nationkey,
    ajm(orders.rowid) as o_ajm,
    ajm(customer.rowid) as c_ajm,
    ajm(supplier.rowid) as s_ajm,
    o_orderkey,
    l_receiptdate,
    l_commitdate,
    l_quantity,
    l_returnflag,
    l_partkey,
    l_suppkey,
    o_orderpriority,
    o_shippriority,
    o_custkey,
    c_mktsegment,

```

```

    l_orderkey,
    l_shipmode,
    p_type,
    ajm(parts.rowid) as p_ajm,
    lineitem.rowid as l_rowid,
    orders.rowid as o_rowid,
    customer.rowid as c_rowid,
    supplier.rowid as s_rowid,
    parts.rowid as p_rowid
FROM
    parts,
    supplier,
    lineitem,
    customer,
    orders

WHERE
    l_orderkey=o_orderkey(+)
    AND
        o_custkey=c_custkey(+)
    AND
        l_suppkey=s_suppkey(+)
    AND
        l_partkey=p_partkey(+)
    order by o_orderkey
;
alter table mj_plocs parallel (degree 160);
}
*sql
{
connect tpcd/tpcd;
drop materialized view log on mj_plocs;
create materialized view log on mj_plocs
tablespace ts_mjlog
storage (initial 32k next 5m pctincrease 0)
partition by range (l_shipdate)
(
partition item1 values less than (to_date('1992-03-01','YYYY-MM-DD'))
partition item2 values less than (to_date('1992-03-24','YYYY-MM-DD'))
partition item3 values less than (to_date('1992-04-14','YYYY-MM-DD'))
partition item4 values less than (to_date('1992-05-01','YYYY-MM-DD'))
partition item5 values less than (to_date('1992-05-16','YYYY-MM-DD'))
partition item6 values less than (to_date('1992-06-01','YYYY-MM-DD'))
partition item7 values less than (to_date('1992-06-16','YYYY-MM-DD'))
partition item8 values less than (to_date('1992-07-01','YYYY-MM-DD'))
partition item9 values less than (to_date('1992-07-16','YYYY-MM-DD'))
partition item10 values less than (to_date('1992-08-01','YYYY-MM-DD'))
partition item11 values less than (to_date('1992-08-16','YYYY-MM-DD'))
partition item12 values less than (to_date('1992-09-01','YYYY-MM-DD'))
partition item13 values less than (to_date('1992-09-16','YYYY-MM-DD'))
partition item14 values less than (to_date('1992-10-01','YYYY-MM-DD'))
partition item15 values less than (to_date('1992-10-16','YYYY-MM-DD'))
partition item16 values less than (to_date('1992-11-01','YYYY-MM-DD'))
partition item17 values less than (to_date('1992-11-16','YYYY-MM-DD'))
partition item18 values less than (to_date('1992-12-01','YYYY-MM-DD'))
partition item19 values less than (to_date('1992-12-16','YYYY-MM-DD'))
partition item20 values less than (to_date('1993-01-01','YYYY-MM-DD'))
partition item21 values less than (to_date('1993-01-16','YYYY-MM-DD'))
partition item22 values less than (to_date('1993-02-01','YYYY-MM-DD'))
partition item23 values less than (to_date('1993-02-16','YYYY-MM-DD'))
partition item24 values less than (to_date('1993-03-01','YYYY-MM-DD'))
partition item25 values less than (to_date('1993-03-16','YYYY-MM-DD'))
partition item26 values less than (to_date('1993-04-01','YYYY-MM-DD'))
partition item27 values less than (to_date('1993-04-16','YYYY-MM-DD'))
partition item28 values less than (to_date('1993-05-01','YYYY-MM-DD'))
partition item29 values less than (to_date('1993-05-16','YYYY-MM-DD'))
partition item30 values less than (to_date('1993-06-01','YYYY-MM-DD'))
partition item31 values less than (to_date('1993-06-16','YYYY-MM-DD'))
partition item32 values less than (to_date('1993-07-01','YYYY-MM-DD'))
partition item33 values less than (to_date('1993-07-16','YYYY-MM-DD'))
partition item34 values less than (to_date('1993-08-01','YYYY-MM-DD'))
partition item35 values less than (to_date('1993-08-16','YYYY-MM-DD'))
partition item36 values less than (to_date('1993-09-01','YYYY-MM-DD'))
partition item37 values less than (to_date('1993-09-16','YYYY-MM-DD'))
partition item38 values less than (to_date('1993-10-01','YYYY-MM-DD'))

```



```

,
partition item150 values less than (to_date('1998-06-01','YYYY-MM-DD'))
,
partition item151 values less than (to_date('1998-06-16','YYYY-MM-DD'))
,
partition item152 values less than (to_date('1998-07-01','YYYY-MM-DD'))
,
partition item153 values less than (to_date('1998-07-16','YYYY-MM-DD'))
,
partition item154 values less than (to_date('1998-08-01','YYYY-MM-DD'))
,
partition item155 values less than (to_date('1998-08-16','YYYY-MM-DD'))
,
partition item156 values less than (to_date('1998-09-01','YYYY-MM-DD'))
,
partition item157 values less than (to_date('1998-09-16','YYYY-MM-DD'))
,
partition item158 values less than (to_date('1998-10-01','YYYY-MM-DD'))
,
partition item159 values less than (to_date('1998-11-01','YYYY-MM-DD'))
,
partition item160 values less than (MAXVALUE)
)
with rowid (c_nationkey, c_custkey, l_returnflag, s_nationkey, o_orderdate,
o_ajm, c_ajm, s_ajm, l_extendedprice, l_discount,
l_shipdate, p_type, p_ajm)
including new values
;
}
*sql
{
connect tpcd/tpcd;
drop materialized view mj_ppssrn;
alter session set hash_area_size = 50000000;
create materialized view mj_ppssrn
pctfree 0
tablespace ts_mj_ppssrn
storage (initial 32k next 31m maxextents unlimited pctincrease 0)
parallel (degree 160)
nologging
partition by range (r_name)
subpartition by hash(p_size)
subpartitions 64
(
partition mj_ppssrn1 values less than ('AMERICA')
,
partition mj_ppssrn2 values less than ('ASIA')
,
partition mj_ppssrn3 values less than ('EUROPE')
,
partition mj_ppssrn4 values less than ('MIDDLE EAST')
,
partition mj_ppssrn5 values less than (MAXVALUE)
)
enable row movement
BUILD immediate
REFRESH fast
ON commit
enable query rewrite
as select
n_name,
ajm(n.rowid) as n_ajm,
p_brand,
p_mfg,
p_name,
p_size,
p_type,
p_partkey,
ajm(p.rowid) as p_ajm,
ps_availqty,
ps_partkey,
ps_suppkey,
ps_supplycost,
s_suppkey,
ajm(s.rowid) as s_ajm,
ajm(r.rowid) as r_ajm,
r_name ,
p.rowid as p_rowid,
ps.rowid as ps_rowid,
s.rowid as s_rowid,
n.rowid as n_rowid,
r.rowid as r_rowid
FROM
partsupp ps,
supplier s,
nation n,
region r,
parts p
WHERE
ps_partkey = p_partkey(+)
AND
ps_suppkey = s_suppkey(+)
AND
s_nationkey = n_nationkey(+)
AND
n_regionkey = r_regionkey(+)
;
}
*sql
{
connect tpcd/tpcd;
drop materialized view ma_plocs_1;
create materialized view ma_plocs_1
pctfree 2
tablespace ts_ma_lineitem
storage (initial 512k next 512k pctincrease 0)
nologging
REFRESH fast
ON commit
enable query rewrite
as

```

```

select
dfy(l_shipdate) as ship_dfy,
dfy(o_orderdate) as order_dfy,
c_nationkey,
s_nationkey,
o_ajm,
c_ajm,
s_ajm,
sum(l_extendedprice * (1 - l_discount)) as volume,
count(l_extendedprice * (1 - l_discount)) as count_volume,
count(*) as count_grp
from
mj_plocs
group by
dfy(l_shipdate),
dfy(o_orderdate),
c_nationkey,
s_nationkey,
o_ajm,
c_ajm,
s_ajm
;
}
*sql
{
connect tpcd/tpcd;
drop materialized view ma_plocs_2;
create materialized view ma_plocs_2
pctfree 2
tablespace ts_ma_lineitem
storage (initial 512k next 512k pctincrease 0)
nologging
REFRESH fast
ON commit
enable query rewrite
as
select
dfm(l_shipdate) as ship_dfm,
p_type,
p_ajm,
sum(l_extendedprice*(1-l_discount)) as type_revenue,
count(l_extendedprice*(1-l_discount)) as count_type_revenue,
count(*) as count_grp
from
mj_plocs
group by
dfm(l_shipdate), p_type, p_ajm
;
}
*wait
*time
{
DONE BUILDING VIEWS
}
*bgoff
%e-scuvo

```

dbgen Changes

driver.c

```

/* Scsid:      @(#)driver.c9.1.1.34      5/1/96  11:45:07 */
/* 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 */
#define LIFENOISE(cnt) if (i % cnt == 0 && verbose) fprintf(stderr, ".")

#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 <errno.h>
#ifdef HP
#include <strings.h>
#endif
#if (defined(WIN32) && !defined(_POSIX_))
#include <process.h>
#include <signal.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 NOSYMETRICS
#define NOTEXTMETRIC
#define NOWH
#define NOCOMM

```

```

#define NOKANJI
#define NOMCX
#include <windows.h>
#pragma warning(default:4201)
#pragma warning(default:4214)
#endif

#include "dss.h"
#include "dstypes.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);
void seed_name(char *tgt, long s, long c, long p);
int partial(int, int);

#ifdef ORIG
#include "part.h"
int partition = 0; /* partition flag */
vals *values;
FILE *in; /* fdes for input file */
int *out; /* output fd array */
FILE *par; /* fdes for params file */
char param[FLEN];
extern char outname[];
extern FILE* outparam; /* fd for output parameter file */
int use_outparam = 0; /* flag to designate whether to use outparam file */
long numpart = 0; /* number of partitions */
long partnum; /* partition number */
int numcol = 0; /* number of columns */
char delim = '|'; /* flatfile delimiter */
#endif /* !ORIG */

extern int optind,
opterr;
extern char *optarg;
long rowcnt = 0,
minrow = 0,
upd_num = 0;
double flt_scale;
#if (defined(WIN32)&&!defined(_POSIX_))
char *spawn_args[25];
#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_time(dss_time_t * t, 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_time(dss_time_t * t, 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);
long sd_order_line(long skip_count);
long sd_part_psupp(long skip_count);
long sd_nation(long skip_count);
long sd_region(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_time(FILE * f);
int hd_nation(FILE * f);
int hd_region(FILE * f);

tdef tdefs[] =
{
{"part.tbl", "part table", 200000, hd_part,
{pr_part, ld_part}, sd_part, NONE},
{"partsupp.tbl", "partsupplier table", 200000, hd_psupp,
{pr_psupp, ld_psupp}, sd_psupp, NONE},
{"supplier.tbl", "suppliers table", 10000, hd_supp,
{pr_supp, ld_supp}, sd_supp, NONE},
{"customer.tbl", "customers table", 150000, hd_cust,
{pr_cust, ld_cust}, sd_cust, NONE},
{"order.tbl", "order table", 150000, hd_order,
{pr_order, ld_order}, sd_order, NONE},
{"lineitem.tbl", "lineitem table", 150000, hd_line,
{pr_line, ld_line}, sd_line, NONE},
{"order.tbl", "order/lineitem tables", 150000, hd_order_line,
{pr_order_line, ld_order_line}, sd_order_line, LINE},
{"part.tbl", "part/partsupplier tables", 200000, hd_part_psupp,
{pr_part_psupp, ld_part_psupp}, sd_part_psupp, PSUPP},
{"time.tbl", "time table", 2557, hd_time,
{pr_time, ld_time}, NO_LFUNC, NONE},
{"nation.tbl", "nation table", NATIONS_MAX, hd_nation,
{pr_nation, ld_nation}, sd_nation, NONE},
{"region.tbl", "region table", NATIONS_MAX, hd_region,
{pr_region, ld_region}, sd_region, NONE},
{"TPCDSSEED", NULL, 0, NO_FUNC,
{NO_FUNC, NO_FUNC}, NO_LFUNC, NONE}
};

int pids[MAX_CHILDREN];

void
mk_sparse(long *low_res, long *high_res, long base, long seq)
{
long low_mask, seq_mask, overflow = 0;
int count = 0;

low_mask = (1<<SPARSE_KEEP) - 1;
seq_mask = (1<<SPARSE_BITS) - 1;
bin_bcd2(base, low_res, high_res);
bcd2_div(low_res, high_res, 1 << SPARSE_KEEP);
bcd2_mul(low_res, high_res, 1 << SPARSE_BITS);
bcd2_add(low_res, high_res, seq);
bcd2_mul(low_res, high_res, 1 << SPARSE_KEEP);
bcd2_add(low_res, high_res, base & low_mask);
bcd2_bin(&low_mask, *low_res);
bcd2_bin(&seq_mask, *high_res);
*low_res = low_mask;
*high_res = seq_mask;
return;
}

/*
 * routines to handle the graceful cleanup of multi-process loads
 */

void
stop_proc(int signum)
{
exit(0);
}

void
kill_load(void)
{
int i;

#if !defined(U2200) && !defined(DOS)
for (i=0; i< MAX_CHILDREN; i++)
if (pids[i])
KILL(pids[i]);
#endif /* !U2200 && !DOS */
return;
}

/*
 * 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
        {
            printf("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);
}

/*
 * generate a particular table
 */
void
gen_tbl(int tnum, long start, long count, long upd_num)
{
    long i;
    order_t o;
    supplier_t supp;
    customer_t cust;
    part_t part;
    dss_time_t t;
    code_t code;
    long sk_low, sk_high;
    static long max_easy = LONG_MAX >> SPARSE_BITS;
    static int completed = 0;

    for (i = start; count; count--, i++)
    {
        LIFENOISE(1000);
        switch(tnum)
        {
            case LINE:
            case ORDER:
            case ORDER_LINE:
                if (i > max_easy)
                    mk_sparse(&sk_low, &sk_high, i,
                        (upd_num == 0)?0:1 + upd_num/(10000 / refresh));
                else
                {
                    sk_low = MK_SPARSE(i,
                        (upd_num == 0)?0:1 + upd_num/(10000 / refresh));
                    sk_high = 0;
                }
                mk_order(sk_low, sk_high, &o);
                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 TIME:
                mk_time(i, &t);
                tdefs[tnum].loader[direct] (&t, 0);
                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;
}
}
completed |= 1 << tnum;
}

void
usage(void)
{
    fprintf(stderr, "%s\\n\\t%s\\n%s %s\\n\\n",
        "USAGE:",
        "dbgen [-{v|f|D}] [-O {fhmt}] [-T {pcso|PSOL}]",
        "[-s <scale>] [-C <procs>] [-S <step>]",
        "dbgen [-v] [-O {dhmt}] [-s <scale>]",
        "[-U <updates>] [-r <percent>]");
    fprintf(stderr, "-C <n> -- use <n> processes to generate data\\n");
    fprintf(stderr, "[-S <n>] -- Under DOS, must be used with -S\\n");
    fprintf(stderr, "-D -- do database load in line\\n");
    fprintf(stderr, "-f -- force. Overwrite existing files\\n");
    fprintf(stderr, "-F -- generate flat files output\\n");
    fprintf(stderr, "-h -- display this message\\n");
    fprintf(stderr, "-n <s> -- inline load into database <s>\\n");
    fprintf(stderr, "-O d -- generate SQL syntax for deletes\\n");
    fprintf(stderr, "-O f -- over-ride default output file names\\n");
    fprintf(stderr, "-O h -- output files with headers\\n");
    fprintf(stderr, "-O m -- produce columnar output\\n");
    fprintf(stderr, "-O r -- generate key ranges for deletes.\\n");
    fprintf(stderr, "-O s -- generate seed sets ONLY\\n");
    fprintf(stderr, "-O t -- use TIME table and julian dates\\n");
    fprintf(stderr, "-r <n> -- updates refresh (n/100)% of the\\n");
    fprintf(stderr, "data set\\n");
    fprintf(stderr, "-R <n> -- resume seed rfile generation with step <n>\\n");
    fprintf(stderr, "-s <n> -- set Scale Factor (SF) to <n>\\n");
    fprintf(stderr, "-S <n> -- build the <n>-th step of the data set\\n");
    fprintf(stderr, "-T c -- generate customers ONLY\\n");
    fprintf(stderr, "-T l -- generate nation/region ONLY\\n");
    fprintf(stderr, "-T L -- generate lineitem ONLY\\n");
    fprintf(stderr, "-T n -- generate nation ONLY\\n");
    fprintf(stderr, "-T o -- generate orders ONLY\\n");
    fprintf(stderr, "-T P -- generate parts/partsupp ONLY\\n");
    fprintf(stderr, "-T p -- generate parts ONLY\\n");
    fprintf(stderr, "-T r -- generate region ONLY\\n");
    fprintf(stderr, "-T s -- generate suppliers ONLY\\n");
    fprintf(stderr, "-T S -- generate partsupp ONLY\\n");
    fprintf(stderr, "-U <s> -- generate <s> update sets\\n");
    fprintf(stderr, "-v -- enable VERBOSE mode\\n");
}

#ifdef ORIG
    fprintf(stderr, "\\n=====\\n");
    fprintf(stderr, "Additions for partitioning:\\n");
    fprintf(stderr, "-p -- enables partitioning\\n");
    fprintf(stderr, "-i <param file> -- input parameter file\\n");
    fprintf(stderr, "-a <out list> -- output prefix list\\n");
    fprintf(stderr, "-o <out prefix> -- output file prefix, -a option overrides -o option\\n");
    fprintf(stderr, "Format of parameter file\\n");
    fprintf(stderr, "<delimiter>\\n");
    fprintf(stderr, "<number of partitions>\\n");
    fprintf(stderr, "<coll type>\\n");
    fprintf(stderr, "<coll pos> - not used at this point\\n");
    fprintf(stderr, "<coll val1>\\n");
    fprintf(stderr, "<coll val2>\\n");
    fprintf(stderr, "...\\n");
    fprintf(stderr, "<coll valm>\\n");
    fprintf(stderr, "Notes:\\n");
    fprintf(stderr, "$s\\n",
        "Column types are: 0 for numbers, 1 for strings, 2 for dates");
    fprintf(stderr, "The direct load option is automatically turned on.\\n");
    fprintf(stderr, "=====\\n");
}

#endif /* ORIG */
fprintf(stderr,
    "\\nTo generate the SF=1 (1GB) database population, use:\\n");
fprintf(stderr, "\\tdbgen -vF -s 1\\n");
fprintf(stderr, "\\n%s %s\\n",
    "To generate the qualification database population",
    "(100 MB), use:\\n");
fprintf(stderr, "\\tdbgen -vF -s 0.1\\n");
fprintf(stderr, "\\nTo generate updates for a SF=1 (1GB), use:\\n");
fprintf(stderr, "\\tdbgen -v -O s -s 1\\n");
fprintf(stderr, "\\tdbgen -v -U 1 -s 1\\n");
}

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

    rowcnt = tdefs[tbl].base * scale;
    if (rowcnt % children)
    {
        fprintf(stderr, "'-' 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);
    }
    if (load_state(scale, children, s))
    {
        seed_name(fname, scale, children, s);
        fprintf(stderr, "Unable to load seeds (%s)\n", fname);
        exit(-1);
    }
    minrow = rowcnt * s + 1;
    if (direct == 0)
        set_files(tbl, s);
    gen_tbl(tbl, minrow, rowcnt, upd_num);
    if (verbose)
        fprintf(stderr, "done.\n");

    return(0);
}

int
pload(int tbl)
{
    int c = 0,
        i,
        status;
    char cmdline[256];

    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 %d children to load %s",
            children, tdefs[tbl].comment);
    }
    for (c=0; c < children; c++)
    {
        pids[c] = SPAWN();
        if (pids[c] == -1)
        {
            perror("Child loader not created");
            kill_load();
            exit(-1);
        }
        else if (pids[c] == 0) /* CHILD */
        {
            SET_HANDLER(stop_proc);
            verbose = 0;
            partial(tbl, c);
            exit(0);
        }
        else if (verbose) /* PARENT */
            fprintf(stderr, ".");
    }
    if (verbose)
        fprintf(stderr, "waiting...");

    c = children;
    while (c)
    {
        i = WAIT(&status, pids[c - 1]);
        if (i == -1 && children)
        {
            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");
        }
        if (status & 0xFF)
        {
            if (status & 0xFF == 0117)
                printf("Process %d: STOPPED\n", i);
            else
                printf("Process %d: rcvd signal %d\n",
                    i, status&0x7F);
        }
        c--;
    }
    if (direct == 0)
    {
        sprintf(cmdline, "mv %s/pl%d.%d%c %s/%s",
            env_config(PATH_TAG, PATH_DFLT), pids[0], tbl,
            'B' + tdefs[tbl].child,
            env_config(PATH_TAG, PATH_DFLT), tdefs[tbl].name);
        system(cmdline);
    }
    #if (defined(WIN32)&&!defined(_POSIX_))
        sprintf(cmdline, "COPY %s\\pl*.%d %s\\%s",
            env_config(PATH_TAG, PATH_DFLT), tbl,
            env_config(PATH_TAG, PATH_DFLT), tdefs[tbl].name);
        fprintf(stderr, "%s\n", cmdline);
        system(cmdline);

        sprintf(cmdline, "DEL %s\\pl*.%d",
            env_config(PATH_TAG, PATH_DFLT), tbl);
        fprintf(stderr, "%s\n", cmdline);
        system(cmdline);
    #else
        for (c=1; c < children; c++)
        {
            sprintf(cmdline,
                "cat %s/pl%d.%d%c >> %s/%s; rm %s/pl%d.%d%c",
                env_config(PATH_TAG, PATH_DFLT), pids[c], tbl,
                'B' + tdefs[tbl].child,
                env_config(PATH_TAG, PATH_DFLT), tdefs[tbl].name,
                env_config(PATH_TAG, PATH_DFLT), pids[c], tbl,
                'B' + tdefs[tbl].child);
            system(cmdline);
        }
    #endif /* WIN32 */

    fprintf(stderr, "done\n");
    return(0);
}

void
process_options(int count, char** vector)
{
    int option;

    #ifndef ORIG
        /* set some default */
        if (partition) {
            (void) sprintf(param, "%s", "/part.param");
            (void) sprintf(outname, "%s", "/dev/sqlldr");
        }
    #endif /* !ORIG */

    #ifndef DOS
    #ifndef ORIG
        while ((option = getopt(count, vector,
            "C:DFfh:O:P:r:R:s:S:T:U:vi:o:a:p")) != -1)
        #else
        while ((option = getopt(count, vector,
            "C:DFfh:O:P:r:R:s:S:T:U:v")) != -1)
        #endif /* !ORIG */
        #else
        while ((option = getopt(count, vector,
            "DFfh:O:P:r:R:s:S:T:U:v")) != -1)
        #endif /* !DOS */
        switch (option)
        {
            #ifndef ORIG
                case 'p':
                    partition = 1;
                    direct = 1;
                    break;
                case 'i':
                    (void) strcpy(param, optarg);
                    break;
                case 'o':
                    strcpy(outname, optarg);
                    break;
                case 'a':
                    if ((outparam = fopen(optarg, "r")) == NULL) {
                        fprintf(stderr, "Error opening file %s\n", optarg);
                    }
                    use_outparam++;
                    break;
            #endif /* !ORIG */
                case 'S':
                    /* generate a particular STEP */
                    step = atoi(optarg) - 1;
                    break;
                case 'v':
                    /* life noises enabled */
                    verbose = 1;
                    break;
                case 'f':
                    /* blind overwrites; Force */
                    force = 1;
                    break;
                case 'T':
                    /* generate a specific table */
                    switch (*optarg)
                    {
                        case 'c':
                            /* generate customer ONLY */
                            table = 1 << CUST;
                            break;
                        case 'L':
                            /* generate lineitems ONLY */
                            table = 1 << LINE;
                            break;
                        case 'l':
                            /* generate code table ONLY */
                            table = 1 << NATION;
                            table |= 1 << REGION;
                            break;
                        case 'n':
                            /* generate nation table ONLY */
                            table = 1 << NATION;
                            break;
                        case 'O':
                            /* generate orders ONLY */
                            table = 1 << ORDER;
                            break;
                        case 'o':
                            /* generate orders/lineitems ONLY */
                            table = 1 << ORDER_LINE;
                            break;
                        case 'P':
                            /* generate part ONLY */
                            table = 1 << PART;
                            break;
                        case 'p':
                            /* generate part/partsupp ONLY */
                            table = 1 << PART_PSUPP;
                            break;
                        case 'r':
                            /* generate region table ONLY */
                            table = 1 << REGION;
                            break;
                        case 'S':
                            /* generate partsupp ONLY */
                            table = 1 << PSUPP;
                            break;
                        case 's':
                            /* generate suppliers ONLY */
                            table = 1 << SUPP;
                            break;
                        case 't':
                            /* generate time ONLY */
                            table = 1 << TIME;
                            break;
                        default:
                    }
                }
    }
}

```

```

        fprintf(stderr, "Unknown table name %s\n",
            optarg);
        usage();
        exit(1);
    }
    break;
case 's': /* scale by Percentage of base rowcount */
case 'P': /* for backward compatibility */
    flt_scale = atof(optarg);
    if (flt_scale < MIN_SCALE)
    {
        int i;

        scale = 1;
        for (i=PART; i < TIME; i++)
        {
            tdefs[i].base *= flt_scale;
            if (tdefs[i].base < 1)
                tdefs[i].base = 1;
        }
    }
    else
        scale = (long)flt_scale;
#ifdef NO_SUPPORT
    if (scale > MAX_SCALE)
    {
        fprintf(stderr, "%s %5.0f %s\n\t%s\n\n",
            "NOTE: Data generation for scale factors >",
            MAX_SCALE,
            "GB is still in development.",
            "and is not yet supported.\n");
        exit(1);
    }
#endif /* NO_SUPPORT */
    break;
case 'O': /* optional actions */
    switch (tolower(*optarg))
    {
        case 'd': /* generate SQL for deletes */
            gen_sql = 1;
            break;
        case 'f': /* over-ride default file names */
            fnames = 1;
            break;
        case 'h': /* generate headers */
            header = 1;
            break;
        case 'm': /* generate columnar output */
            columnar = 1;
            break;
        case 'r': /* generate key ranges for delete */
            gen_rng = 1;
            break;
        case 's': /* generate seed sets */
            bld_seeds = 1;
            break;
        case 't': /* use TIME table and join fields */
            oldtime = 1;
            if (updates == 0)
                table |= (1 << TIME);
            break;
        default:
            fprintf(stderr, "Unknown option name %s\n",
                optarg);
            usage();
            exit(1);
    }
    break;
case 'D': /* direct load of generated data */
    direct = 1;
    break;
case 'F': /* generate flat files for later loading */
    direct = 0;
    break;
case 'U': /* generate flat files for update stream */
    updates = atoi(optarg);
    break;
case 'r': /* set the refresh (update) percentage */
    refresh = atoi(optarg);
    break;
case 'R': /* resume seed file generation */
    resume = atoi(optarg);
    break;
#ifdef DOS
case 'C':
    children = atoi(optarg);
    if (children > MAX_CHILDREN)
    {
        fprintf(stderr, "Children reset to limit of %d\n",
            MAX_CHILDREN);
        children = MAX_CHILDREN;
    }
    break;
#endif /* !DOS */
case 'n': /* set name of database for direct load */
    db_name = (char *)malloc(strlen(optarg) + 1);
    MALLOC_CHECK(db_name);
    strcpy(db_name, optarg);
    break;
default:
    printf("ERROR: option '%c' unknown.\n",
        *(vector[optind] + 1));
case 'h': /* something unexpected */
    fprintf(stderr,
        "TPC-D Population Generator (Version %d.%d.%d)\n",
        VERSION, RELEASE,
        MODIFICATION, PATCH);
    fprintf(stderr, "Copyright %s %s\n", TPC, C_DATES);
    usage();
    exit(1);
}

return;
}

/*
 * MAIN
 *
 * assumes the existence of getopt() to clean up the command
 * line handling
 */
int
main(int ac, char **av)
{
    int i,
        C;

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

    force = 0;
    verbose = 0;
    columnar = 0;
    bld_seeds = 0;
    header = 0;
    oldtime = 0;
    direct = 0;
    scale = 1;
    flt_scale = 1.0;
    updates = 0;
    refresh = UPD_PCT;
    resume = -1;
    step = -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 */
    fnames = 0;
    db_name = NULL;
    gen_sql = 0;
    gen_rng = 0;
    children = 1;

    process_options(ac, av);

#ifdef ORIG
    if (partition) {
        if ((par = fopen(param, "r")) == NULL) {
            fprintf(stderr, "Unable to open file '%s'.\n", param);
            fprintf(stderr, "%s: %s\n", param, strerror(errno));
            exit(-1);
        }
        (void) process_param_file();
    }
#endif /* ORIG */

    if (defined(WIN32) && !defined(_POSIX_))
        for (i=0; i < ac; i++)
        {
            spawn_args[i] = malloc((strlen(av[i]) + 1) * sizeof(char));
            MALLOC_CHECK(spawn_args[i]);
            strcpy(spawn_args[i], av[i]);
        }
        spawn_args[ac] = NULL;
    #endif

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

    load_dists();
    /* have to do this after init */
    tdefs[NATION].base = nations.count;
    tdefs[REGION].base = regions.count;

    /*
     * updates are never parallelized
     */
    if (updates)
    {
        for (i=LINE; i <= ORDER_LINE; i++)
        {
            if (table & (1 << i))
            {
                if (load_state(scale, children, children))
                {
                    fprintf(stderr, "Unable to load seeds (%d)\n",
                        scale);
                    exit(-1);
                }
                rowcnt = tdefs[i].base / 10000 * scale * refresh;
                upd_num = 0;
                while (upd_num < updates)
                {
                    minrow = rowcnt * upd_num + 1;
                    if (verbose)
                        fprintf(stderr,
                            "Generating updates for %s [pid: %d]",
                            tdefs[i].comment, DSS_PROC);
                    gen_tbl(i, minrow, rowcnt, upd_num + 1);
                    if (verbose)
                        fprintf(stderr, "done.\n");
                    pr_drange(i, minrow, rowcnt, upd_num + 1);
                    upd_num++;
                }
            }
        }
    }
}

```

```

        exit(0);
    }
}
/*
 * seed gen is never parallelized
 */
if (bld_seeds)
{
    if (resume== -1)
    {
        c=1;
    }
    else
    {
        c=resume;
    }
    if (load_state(scale, children, c-1))
    {
        char fname[80];
        seed_name(fname, scale, children, c-1);
        fprintf(stderr, "Unable to load seeds (%s)\n", fname);
        exit(-1);
    }
}

for (; c <= children; c++)
{
    for (i=PART; i <= PART_PSUPP; i++)
    {
        if (table & (1 << i))
        {
            if ((tdefs[i].base * scale % children) == 0)
            {
                rowcnt = tdefs[i].base * scale / children;
            }
            else
            {
                fprintf(stderr, "Load cannot be evenly split!");
                exit(1);
            }
            if (verbose)
            {
                fprintf(stderr,
                    "Generating seeds for %s [step: %d] ",
                    tdefs[i].comment, c);
                tdefs[i].gen_seed(rowcnt);
                if (verbose)
                {
                    fprintf(stderr, "done.\n");
                }
            }
            if (store_state(scale, children, c))
            {
                fprintf(stderr, "Unable to store seeds\n");
                exit(-1);
            }
        }
    }
    exit(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);
                }
            }
        }
    }

    for (i = PART; i <= REGION; i++)
    {
        if (table & (1 << i))
        {
            if (children > 1 && i < TIME)
            {
                if (step >= 0)
                {
                    partial(i, step);
                }
            }
            else
            {
                fprintf(stderr,
                    "Parallel load is not supported on your platform.\n");
                exit(1);
            }
        }
    }
}

#ifdef DOS
    else
    {
        fprintf(stderr,
            "Parallel load is not supported on your platform.\n");
        exit(1);
    }
}

#else
    else
    {
        pload(i);
    }
}

#endif /* DOS */
else
{
    minrow = 1;
    if (i < TIME)
    {
        rowcnt = tdefs[i].base * scale;
    }
    else
    {
        rowcnt = tdefs[i].base;
    }
    if (verbose)
    {
        fprintf(stderr, "Generating data for %s [pid: %d]",
            tdefs[i].comment, DSS_PROC);
        gen_tbl(i, minrow, rowcnt, upd_num);
        if (verbose)
        {
            fprintf(stderr, "done.\n");
        }
    }
}

if (direct)
    close_direct();

return (0);
}

```

load_stub.c

```

/*
 * Title:      load_stub.c
 * Scsid:      @(#)load_stub.c9.1.1.11      9/7/95  16:24:05
 * Description: stub routines for:
 *               inline load of dss benchmark
 *               header creation for dss benchmark
 */
#include <stdio.h>
#include "config.h"
#include "dss.h"
#include "dstypes.h"
#include "part.h"

extern vals *values;
extern int *out; /* output fd array */
extern long partnum; /* partition number */
extern int partition;
char tmpdate[16]; /* temporary storage to store dates */
extern void Exitprog();

int
close_direct(void)
{
    /* any post load cleanup goes here */
    return(0);
}

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

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

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

    return(0);
}

int
ld_cust (customer_t *c, int mode)
{
    char line[512];
    char *ptr;

    if (partition) {
        partnum = find_partition(c->mktsegment, 1);
    }
    PPR_STRT(line, ptr);
    PPR_INT(ptr, c->custkey);
    PPR_VSTR(ptr, c->name, C_NAME_LEN);
    PPR_VSTR(ptr, c->address,
        (columnar)?(long)(ceil(C_ADDR_LEN * V_STR_HGH)):c->alen);
    PPR_INT(ptr, c->nation_code);
    PPR_STR(ptr, c->phone, PHONE_LEN);
    PPR_MONEY(ptr, c->sacctbal);
    PPR_STR(ptr, c->mktsegment, C_MSEG_LEN);
    PPR_VSTR(ptr, c->comment,
        (columnar)?(long)(ceil(C_CMNT_LEN * V_STR_HGH)):c->clen);
    PPR_END(ptr);

    if (write(out[partnum], line, strlen(line)) == -1) {
        fprintf(stderr, "Error when writing customer data\n");
        Exitprog(errno);
    }

    return(0);
}

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

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

    return(0);
}

int
ld_part (part_t *pp, int mode)
{
    char line[512];
    char *ptr;
    double tmp = pp->size;

    if (partition) {
        partnum = find_partition(&(tmp), 0);
    }

    PPR_STRT(line, ptr);
    PPR_INT(ptr, pp->partkey);
    PPR_VSTR(ptr, pp->name,
        (columnar)?(long)P_NAME_LEN:pp->nlen);
    PPR_STR(ptr, pp->mfg, P_MFG_LEN);
    PPR_STR(ptr, pp->brand, P_BRND_LEN);
}

```

```

PPR_VSTR(ptr, pp->type,
(columnar)?(long)P_TYPE_LEN:pp->tlen);
PPR_INT(ptr, pp->size);
PPR_STR(ptr, pp->container, P_CNTR_LEN);
PPR_MONEY(ptr, pp->retailprice);
PPR_VSTR(ptr, pp->comment,
(columnar)?(long)(ceil(P_CMNT_LEN * V_STR_HGH)):pp->crlen);
PPR_END(ptr);

if (write(out[partnum], line, strlen(line)) == -1) {
    fprintf(stderr, "Error when writing parts data\n");
    Exitprog(errno);
}

return(0);
}

int
ld_psupp (part_t *pp, int mode)
{
    static int count = 0;

    if (! count++)
        printf("%s %s\n",
            "No load routine has been defined for the",
            "psupp table\n");

    return(0);
}

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

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

    return(0);
}

int
ld_time (dss_time_t *tp, int mode)
{
    static int count = 0;

    if (! count++)
        printf("No load routine has been defined for the time table\n");

    return(0);
}

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

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

    return(0);
}

int
ld_supp (supplier_t *sp, int mode)
{
    static int count = 0;

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

    return(0);
}

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

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

    return(0);
}

int
ld_order (order_t *o, int mode)
{
    char line[512];
    char *ptr;
    static int last_mode = 0;

    if (mode != last_mode)
        last_mode = mode;

    if (partition) {
        (void) strcpy(tmpdate, o->odate);
        partnum = find_partition(tmpdate, 2);
    }
    PPR_STRT(line, ptr);
    PPR_BCD2(ptr, o->okey_high, o->okey_low);
    PPR_INT(ptr, o->custkey);
    PPR_CHR(ptr, o->orderstatus);
    PPR_MONEY(ptr, o->totalprice);
    PPR_STR(ptr, o->odate, DATE_LEN);

    PPR_STR(ptr, o->opriority, O_OPPIO_LEN);
    PPR_STR(ptr, o->clerk, O_CLRK_LEN);
    PPR_INT(ptr, o->spriority);
    PPR_VSTR(ptr, o->comment,
(columnar)?(long)(ceil(O_CMNT_LEN * V_STR_HGH)):o->crlen);
    PPR_END(ptr);

    if (write(out[partnum], line, strlen(line)) == -1) {
        fprintf(stderr, "Error when writing order data\n");
        Exitprog(errno);
    }

    return(0);
}

ld_line (order_t *o, int mode)
{
    char line[512];
    char *ptr;
    static int last_mode = 0;
    long i;

    if (mode != last_mode)
        last_mode = mode;

    for (i = 0; i < o->lines; i++) {
        if (partition) {
            (void) strcpy(tmpdate, o->l[i].sdate);
            partnum = find_partition(tmpdate, 2);
        }
        PPR_STRT(line, ptr);
        PPR_BCD2(ptr, o->l[i].okey_high, o->l[i].okey_low);
        PPR_INT(ptr, o->l[i].partkey);
        PPR_INT(ptr, o->l[i].suppkey);
        PPR_INT(ptr, o->l[i].lcnt);
        PPR_INT(ptr, o->l[i].quantity);
        PPR_MONEY(ptr, o->l[i].eprice);
        PPR_MONEY(ptr, o->l[i].discount);
        PPR_MONEY(ptr, o->l[i].tax);
        PPR_CHR(ptr, o->l[i].rflag[0]);
        PPR_CHR(ptr, o->l[i].lstatus[0]);
        PPR_STR(ptr, o->l[i].sdate, DATE_LEN);
        PPR_STR(ptr, o->l[i].cdate, DATE_LEN);
        PPR_STR(ptr, o->l[i].rdate, DATE_LEN);
        PPR_STR(ptr, o->l[i].shipinstruct, L_INST_LEN);
        PPR_STR(ptr, o->l[i].shipmode, L_SMODE_LEN);
        PPR_VSTR(ptr, o->l[i].comment,
(columnar)?(long)(ceil(L_CMNT_LEN * V_STR_HGH)):o->l[i].crlen);
        PPR_END(ptr);
        if (write(out[[partnum]], line, strlen(line)) == -1) {
            fprintf(stderr, "Error when writing lineitem data\n");
            Exitprog(errno);
        }
    }

    return(0);
}

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

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

    return(0);
}

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

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

    return(0);
}

int
ld_nation (code_t *cp, int mode)
{
    static int count = 0;

    if (! count++)
        printf("%s %s\n",
            "No load routine has been defined",
            "for the nation table");

    return(0);
}

int

```

```

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

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

    return(0);
}

int
ld_region (code_t *cp, int mode)
{
    static int count = 0;

    if (! count++)
        printf("%s %s\n",
            "No load routine has been defined",
            "for the region table");

    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);
}

```

part.c

```

/*-----
|      Copyright (c) 1997 Oracle Corp, Redwood Shores, CA
|      ADVANCED TECH PERFORMANCE GROUP
|      All Rights Reserved
|-----
| FILENAME
|      part.c
| DESCRIPTION
|      Functions to support partitioning.
| MODIFIED
|      pswong    03/13/97 - created
|-----*/

```

```

#include "part.h"

extern FILE *par;
extern int *out;
extern char delim;
char outname[FLEN];
extern int use_outparam;
FILE *outparam;
extern long numpart;
extern vals *values;

void process_param_file()
{
    char ofile[128];
    char line[128];
    char *pos, *str;
    int i,j;
    int retcode;
    struct stat fstats;

    /* get delimiter */

    fscanf(par, "%c\n", &delim);

    /* get number of partitions */

    fscanf(par, "%ld\n", &numpart);

    /* malloc some structures */

    values = (vals *) malloc(sizeof(vals));
    out = (int *) malloc(numpart * sizeof(int));

```

```

/* extract the column types */

fscanf(par, "%d\n", &(values->type));

/* extract column position */

fscanf(par, "%d\n", &(values->pos));

/* setup output file descriptors */

for (j=0; j<numpart; j++) {
    if (use_outparam) {
        /* get the file names */
        if (fscanf(outparam, "%s\n", ofile) == NULL) {
            fprintf(stderr, "Error: Number of entries in output parameter file less than the
number of partitions\n");
            (void) Exitprog(-1);
        }
    } else {
        sprintf(ofile, "%s.%d", (char *) outname, (j+1));
    }
    out[j] = open(ofile, O_RDWR|O_APPEND|O_CREAT);
    if (out[j] == -1) {
        fprintf(stderr, "Unable to open file '%s'.\n", ofile);
        fprintf(stderr, "%s: %s\n", ofile, strerror(errno));
        (void) Exitprog(-1);
    }
}

/* allocate memory for the values */

switch(values->type) {
case NUM_TYP:
    values->vals = (double *) malloc(sizeof(double) * numpart);
    break;
case STR_TYP:
    values->vals = (char **) malloc(sizeof(char *) * numpart);
    break;
case DAT_TYP:
    values->vals = (dtyp *) malloc(sizeof(dtyp) * numpart);
    break;
}

/* now read the values */
/* could have easily done by fscanf */

for (j=0; j<numpart; j++) {
    if (fgets(line, 127, par) == NULL) {
        fprintf(stderr, "Error: number of boundaries definition does not equal to
the number of partitions.\n");
        Exitprog(-1);
    }
    str = strtok(line, DELIM);
    switch(values->type) {
    case NUM_TYP:
        ((double *)values->vals)[j] = (double) atof(str);
        break;
    case STR_TYP:
        ((char **)values->vals)[j] = (char *) malloc(strlen(str) + 1);
        strcpy(((char **)values->vals)[j], str);
        break;
    case DAT_TYP:
        ASS_DATE(((dtyp *)values->vals)[j], str);
        break;
    }
}
}

void Exitprog(errno)
{
    int i;

    /* free some memory */

    if (values->type == STR_TYP) {
        for(i=0; i<numpart; i++)
            free(((char **)values->vals)[i]);
    }

    for (i=0; i<numpart; i++)
        close(out[i]);

    free(values->vals);
    free(out);
    free(values);
    exit(errno);
}

int find_partition(cval, type)
{
    void *cval;
    int type;

    {
        dtyp vdat;

        if (type != values->type) {
            fprintf(stderr, "Type mismatch between parameter file and the column type\n");
            Exitprog(-1);
        }
        switch(values->type) {
        case NUM_TYP:
            return (part_search((double *)values->vals, (double *) cval, NUM_TYP,
fltcmp));
        case STR_TYP:
            return (part_search((char **)values->vals, (char *) cval, STR_TYP,
strgcmp));
        case DAT_TYP:
            ASS_DATE(vdat, (char *)cval);

```

```

        return (part_search((dtyp *)values->vals, (dtyp *) &vdat, DAT_TYP,
        datcmp));
    }
}

int fltcmp(a,b)
    void *a;
    void *b;
{
    if (*((double *)a) == *((double *)b))
        return 0;
    else {
        if (*((double *)a) > *((double *)b)) return 1;
        else return -1;
    }
}

int strgcmp(a,b)
    void *a;
    void *b;
{
    int ret;

    ret = strncmp((char *)a, (char *)b,
    MINV(strlen((char *)a), strlen((char *)b)));
    return(ret);
}

int datcmp(a,b)
    void *a;
    void *b;
{
    if (((dtyp *)a)->year > ((dtyp *)b)->year) {
        return 1;
    } else {
        if (((dtyp *)a)->year < ((dtyp *)b)->year) {
            return -1;
        } else {
            if (((dtyp *)a)->month > ((dtyp *)b)->month) {
                return 1;
            } else {
                if (((dtyp *)a)->month < ((dtyp *)b)->month) {
                    return -1;
                } else {
                    if (((dtyp *)a)->day > ((dtyp *)b)->day) {
                        return 1;
                    } else {
                        if (((dtyp *)a)->day < ((dtyp *)b)->day) {
                            return -1;
                        } else {
                            return 0;
                        }
                    }
                }
            }
        }
    }
}

void *ptr_arith(ptr, inc, type)
    void *ptr;
    int inc;
    int type;
{
    switch(type) {
        case NUM_TYP:
            return ((double *)ptr + inc);
            break;
        case STR_TYP:
            return *((char **)ptr + inc);
            break;
        case DAT_TYP:
            return ((dtyp *)ptr + inc);
    }
}

/* Simple search, returns index into the array */
/* arr is the array of size numpart to be searched */
/* item is the item to search for */
/* fp is the pointer to the comparison function */
/* fp(&a,&b) should return 0 if a == b */
/* 1 if a > b */
/* -1 if a < b */

int part_search(arr,item,type,fp)
    void *arr;
    void *item;
    int type;
    int (*fp)();
{
    static int hint = 0; /* hint */
    int i;

    while (1) {
        /* if we are smaller than the very first partition */
        if (hint == 0) {
            if (fp(ptr_arith(arr,0,type),item) > 0)
                return 0;
            else {
                hint++;
                continue;
            }
        }
    }
}

```

```

/* if we are larger than the very last partition */
if (hint == (numpart - 1)) {
    if (fp(ptr_arith(arr,hint,type),item) <= 0)
        return (numpart-1);
    else {
        if (fp(ptr_arith(arr,(hint-1),type),item) <= 0)
            return hint;
        else {
            hint--;
            continue;
        }
    }
}

if (fp(ptr_arith(arr,hint,type),item) > 0) {
    /* let's see if item is in the current partition */
    if (fp(ptr_arith(arr,(hint-1),type),item) <= 0)
        return hint;
    else
        hint--;
} else hint++;
}
}

```

part.h

```

/*-----
      Copyright (c) 1997 Oracle Corp, Redwood Shores, CA
      ADVANCED TECH PERFORMANCE GROUP
      All Rights Reserved
-----*/

FILENAME
part.h
DESCRIPTION
Header file for dbgen partitioning extension.
MODIFIED
pswong 03/13/97 - created
-----*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include <fcntl.h>
#include <sys/types.h>
#include <sys/stat.h>

#define FLEN 128

typedef struct dt {
    int year;
    int month;
    int day;
} dtyp;

typedef struct val {
    int type;
    int pos;
    void *vals;
} vals;

#define ASS_DATE(dat,str) \
{ dat.year = atoi(strtok(str,"-\n")); \
  dat.month = atoi(strtok(NULL,"-\n")); \
  dat.day = atoi(strtok(NULL,"-\n")); }

#define MINV(x,y) ((x < y) ? x : y)

/* some marco to do the print */

#ifndef SEPARATOR
#define SEPARATOR '|' /* field speparator for generated flat files */
#endif

#define PPR_STR(f,str,len) \
if (columnar) sprintf(f, "%-s", len, str); \
else sprintf(f, "%s%c", str, SEPARATOR); \
f += strlen(f)

#define PPR_VSTR(f,str,len) PPR_STR(f,str,len)

#define PPR_INT(f,long) \
if (columnar) sprintf(f, "%12ld", long); \
else sprintf(f, "%ld%c", long, SEPARATOR); \
f += strlen(f)

#define PPR_BCD2(f,high,low) \
if (high == 0) \
if (columnar) sprintf(f, "%12ld", low); \
else sprintf(f, "%ld%c", low, SEPARATOR); \
else \
if (columnar) sprintf(f, "%5ld%07ld", high, low); \
else sprintf(f, "%ld%07ld%c", high, low, SEPARATOR); \
f += strlen(f)

#define PPR_KEY(f,long) \
sprintf(f, "%ld", long); \
f += strlen(f)

#define MONEY_COL_FMT "%12ld.%02ld"
#define PPR MONEY(f,flt) \
if (columnar) sprintf(f, MONEY_COL_FMT, \
flt/100, ((flt < 0)?-flt:flt)*100); \
else sprintf(f, "%ld.%02ld%c", \
flt/100, ((flt < 0)?-flt:flt)*100, SEPARATOR); \
fp += strlen(fp)

```

```

#define PPR_CHR(fp,chr) \
    if (columnar) sprintf(fp, "%c ", chr); \
    else sprintf(fp,"%c%c", chr, SEPARATOR); \
    fp += strlen(fp)

#define PPR_STRT(dest,ptr) (ptr = dest)
#define PPR_END(fp)    sprintf(fp, "\n") /* finish the record here */

#ifdef MDY_DATE
#define PR_DATE(tgt, yr, mn, dy) \
    sprintf(tgt, "%02d-%02d-19%02d", mn, dy, yr)
#else
#define PR_DATE(tgt, yr, mn, dy) \
    sprintf(tgt, "19%02d-%02d-%02d", yr, mn, dy)
#endif /* DATE_FORMAT */

/* Could add a integer type and a character type to allow faster processing. */
#define NUM_TYP 0 /* number type */
#define STR_TYP 1 /* str/char type */
#define DAT_TYP 2 /* date type */

#define DELIM "|\\n"

void Exitprog();
void process_param_file();
int fltcmp();
int datcmp();
int strgcmp();
int part_search();
int find_partition();

```

print.c

```

/* Scosid: @(#)print.c9.1.1.18 12/13/95 10:56:16 */
/* generate flat files for data load */
#include <stdio.h>
#ifdef VMS
#include <sys/types.h>
#endif
#if defined(SUN)
#include <unistd.h>
#endif
#include <math.h>

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

/*
 * Function Prototypes
 */
FILE *print_prep_PROTO((int table, int update));
int pr_drange_PROTO((int tbl, long min, long cnt, long num));

FILE *
print_prep(int table, int update)
{
    char upath[128];
    FILE *res;

    if (updates)
    {
        if (update > 0) /* updates */
        {
            sprintf(upath, "%s/%s.u%d",
                env_config(PATH_TAG, PATH_DFLT),
                tdefs[table].name, update);
        }
        else /* deletes */
        {
            sprintf(upath, "%s/delete.%d",
                env_config(PATH_TAG, PATH_DFLT), -update);
        }
        return(fopen(upath, "w"));
    }
    res = tbl_open(table, "w");
    if (res == NULL)
    {
        fprintf(stderr, "File open failed for %s\n",
            tdefs[table].name);
        exit(-1);
    }
    return(res);
}

int
pr_cust(customer_t *c, int mode)
{
    static FILE *fp = NULL;
    static int last_mode = 0;

    if (fp == NULL)
        fp = print_prep(CUST, 0);

    PR_STRT(fp);
    PR_INT(fp, c->custkey);
    PR_VSTR(fp, c->name, C_NAME_LEN);
    PR_VSTR(fp, c->address,
        (columnar)?(long)(ceil(C_ADDR_LEN * V_STR_HGH)):c->alen);
    PR_INT(fp, c->nation_code);
    PR_STR(fp, c->phone, PHONE_LEN);
    PR_MONEY(fp, c->acctbal);
    PR_STR(fp, c->mktsegment, C_MSEG_LEN);
    PR_VSTR(fp, c->comment,
        (columnar)?(long)(ceil(C_CMNT_LEN * V_STR_HGH)):c->clen);
    PR_END(fp);

    return(0);
}

```

```

/*
 * print the numbered order
 */
int
pr_order(order_t *o, int mode)
{
    static FILE *fp_o = NULL;
    static int last_mode = 0;

    if (fp_o == NULL || mode != last_mode)
    {
        if (fp_o)
            fclose(fp_o);
        fp_o = print_prep(ORDER, mode);
        last_mode = mode;
    }

    PR_STRT(fp_o);
    PR_BCD2(fp_o, o->okey_high, o->okey_low);
    PR_INT(fp_o, o->custkey);
    PR_CHR(fp_o, o->orderstatus);
    PR_MONEY(fp_o, o->totalprice);
    PR_STR(fp_o, o->odate, DATE_LEN);
    PR_STR(fp_o, o->opriority, O_OPRI_LEN);
    PR_STR(fp_o, o->clerk, O_CLRK_LEN);
    PR_INT(fp_o, o->spriority);
    PR_VSTR(fp_o, o->comment,
        (columnar)?(long)(ceil(O_CMNT_LEN * V_STR_HGH)):o->clen);
    PR_END(fp_o);

    return(0);
}

/*
 * print an order's lineitems
 */
int
pr_line(order_t *o, int mode)
{
    static FILE *fp_l = NULL;
    static int last_mode = 0;
    long i;

    if (fp_l == NULL || mode != last_mode)
    {
        if (fp_l)
            fclose(fp_l);
        fp_l = print_prep(LINE, mode);
        last_mode = mode;
    }

    for (i = 0; i < o->lines; i++)
    {
        PR_STRT(fp_l);
        PR_BCD2(fp_l, o->l[i].okey_high, o->l[i].okey_low);
        PR_INT(fp_l, o->l[i].partkey);
        PR_INT(fp_l, o->l[i].suppkey);
        PR_INT(fp_l, o->l[i].lcnt);
        PR_INT(fp_l, o->l[i].quantity);
        PR_MONEY(fp_l, o->l[i].eprice);
        PR_MONEY(fp_l, o->l[i].discount);
        PR_MONEY(fp_l, o->l[i].tax);
        PR_CHR(fp_l, o->l[i].rflag[0]);
        PR_CHR(fp_l, o->l[i].lstatus[0]);
        PR_STR(fp_l, o->l[i].sdate, DATE_LEN);
        PR_STR(fp_l, o->l[i].cdate, DATE_LEN);
        PR_STR(fp_l, o->l[i].rdate, DATE_LEN);
        PR_STR(fp_l, o->l[i].shipinstruct, L_INST_LEN);
        PR_STR(fp_l, o->l[i].shipmode, L_SMODE_LEN);
        PR_VSTR(fp_l, o->l[i].comment,
            (columnar)?(long)(ceil(L_CMNT_LEN * V_STR_HGH)):o->l[i].clen);
        PR_END(fp_l);
    }

    return(0);
}

/*
 * print the numbered order *and* its associated lineitems
 */
int
pr_order_line(order_t *o, int mode)
{
    tdefs[ORDER].name = tdefs[ORDER_LINE].name;
    pr_order(o, mode);
    pr_line(o, mode);

    return(0);
}

/*
 * print the given part
 */
int
pr_part(part_t *part, int mode)
{
    static FILE *p_fp = NULL;

    if (p_fp == NULL)
        p_fp = print_prep(PART, 0);

    PR_STRT(p_fp);
    PR_INT(p_fp, part->partkey);
    PR_VSTR(p_fp, part->name,
        (columnar)?(long)P_NAME_LEN:part->nlen);
    PR_STR(p_fp, part->mfg, P_MFG_LEN);
    PR_STR(p_fp, part->brnd, P_BRND_LEN);
    PR_VSTR(p_fp, part->type,
        (columnar)?(long)P_TYPE_LEN:part->tlen);
    PR_INT(p_fp, part->size);
    PR_STR(p_fp, part->container, P_CNTR_LEN);
}

```

```

PR_MONEY(p_fp, part->retailprice);
PR_VSTR(p_fp, part->comment,
        (columnar)?(long)(ceil(P_CMNT_LEN * V_STR_HGH)):part->crlen);
PR_END(p_fp);

return(0);
}

/*
 * print the given part's suppliers
 */
int
pr_psupp(part_t *part, int mode)
{
    static FILE *ps_fp = NULL;
    long i;

    if (ps_fp == NULL)
        ps_fp = print_prep(PSUPP, mode);

    for (i = 0; i < SUPP_PER_PART; i++)
    {
        PR_STRT(ps_fp);
        PR_INT(ps_fp, part->s[i].partkey);
        PR_INT(ps_fp, part->s[i].supkey);
        PR_INT(ps_fp, part->s[i].qty);
        PR_MONEY(ps_fp, part->s[i].scost);
        PR_VSTR(ps_fp, part->s[i].comment,
                (columnar)?(long)(ceil(PS_CMNT_LEN * V_STR_HGH)):part->s[i].crlen);
        PR_END(ps_fp);
    }

    return(0);
}

/*
 * print the given part *and* its suppliers
 */
int
pr_part_psupp(part_t *part, int mode)
{
    tdefs[PART].name = tdefs[PART_PSUPP].name;
    pr_part(part, mode);
    pr_psupp(part, mode);

    return(0);
}

int
pr_supp(supplier_t *supp, int mode)
{
    static FILE *fp = NULL;

    if (fp == NULL)
        fp = print_prep(SUPP, mode);

    PR_STRT(fp);
    PR_INT(fp, supp->supkey);
    PR_STR(fp, supp->name, S_NAME_LEN);
    PR_VSTR(fp, supp->address,
            (columnar)?(long)(ceil(S_ADDR_LEN * V_STR_HGH)):supp->alen);
    PR_INT(fp, supp->nation_code);
    PR_STR(fp, supp->phone, PHONE_LEN);
    PR_MONEY(fp, supp->acctbal);
    PR_VSTR(fp, supp->comment,
            (columnar)?(long)(ceil(S_CMNT_LEN * V_STR_HGH)):supp->crlen);
    PR_END(fp);

    return(0);
}

int
pr_time(dss_time_t *t, int mode)
{
    static FILE *fp = NULL;

    if (fp == NULL)
        fp = print_prep(TIME, mode);

    PR_STRT(fp);
    PR_INT(fp, t->timekey);
    PR_STR(fp, t->alpha, T_ALPHA_LEN);
    PR_INT(fp, t->year);
    PR_INT(fp, t->month);
    PR_INT(fp, t->week);
    PR_INT(fp, t->day);
    PR_END(fp);

    return(0);
}

int
pr_nation(code_t *c, int mode)
{
    static FILE *fp = NULL;

    if (fp == NULL)
        fp = print_prep(NATION, mode);

    PR_STRT(fp);
    PR_INT(fp, c->code);
    PR_STR(fp, c->text, NATION_LEN);
    PR_INT(fp, c->join);
    PR_VSTR(fp, c->comment,
            (columnar)?(long)(ceil(N_CMNT_LEN * V_STR_HGH)):c->crlen);
    PR_END(fp);

    return(0);
}

int

pr_region(code_t *c, int mode)
{
    static FILE *fp = NULL;

    if (fp == NULL)
        fp = print_prep(REGION, mode);

    PR_STRT(fp);
    PR_INT(fp, c->code);
    PR_STR(fp, c->text, REGION_LEN);
    PR_VSTR(fp, c->comment,
            (columnar)?(long)(ceil(R_CMNT_LEN * V_STR_HGH)):c->crlen);
    PR_END(fp);

    return(0);
}

/*
 * 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
pr_drange(int tbl, long min, long cnt, long num)
{
    static int last_num = 0;
    static FILE *dfp = NULL;
    int child = -1;
    long start, last, new;

    if (last_num != num)
    {
        if (dfp)
            fclose(dfp);
        dfp = print_prep(tbl, -num);
        if (dfp == NULL)
            return(-1);
        last_num = num;
    }

    start = MK_SPARSE(min, num/ (10000 / refresh));
    last = start - 1;
    for (child=min; cnt > 0; child++, cnt--)
    {
        new = MK_SPARSE(child, num/ (10000 / refresh));
        if (gen_rng == 1 && new - last == 1)
        {
            last = new;
            continue;
        }

        if (gen_sql)
        {
            fprintf(dfp,
                "delete from %s where %s between %ld and %ld;\n",
                tdefs[ORDER].name, "o_orderkey", start, last);
            fprintf(dfp,
                "delete from %s where %s between %ld and %ld;\n",
                tdefs[LINE].name, "l_orderkey", start, last);
            fprintf(dfp, "commit work;\n");
        }
        else
        {
            if (gen_rng)
            {
                PR_STRT(dfp);
                PR_INT(dfp, start);
                PR_INT(dfp, last);
                PR_END(dfp);
            }
            else
            {
                PR_STRT(dfp);
                PR_KEY(dfp, new);
                PR_END(dfp);
            }
        }

        start = new;
        last = new;
    }

    if (gen_rng)
    {
        PR_STRT(dfp);
        PR_INT(dfp, start);
        PR_INT(dfp, last);
        PR_END(dfp);
    }

    return(0);
}

```


Appendix C. Query Text & Query Output

Begin Execution at Wed Feb 10 16:38:46 1999

-- using default substitutions
-- Query 1 (Original)

```
SELECT
L_RETURNFLAG,
L_LINESTATUS,
SUM(L_QUANTITY) AS SUM_QTY,
SUM(L_EXTENDEDPRICE) AS SUM_BASE_PRICE,
SUM(L_EXTENDEDPRICE * (1 - L_DISCOUNT)) AS SUM_DISC_PRICE,
SUM(L_EXTENDEDPRICE * (1 - L_DISCOUNT) * (1 + L_TAX)) AS SUM_CHARGE,
AVG(L_QUANTITY) AS AVG_QTY,
AVG(L_EXTENDEDPRICE) AS AVG_PRICE,
AVG(L_DISCOUNT) AS AVG_DISC,
COUNT(*) AS COUNT_ORDER
FROM LINEITEM
WHERE L_SHIPDATE <= TO_DATE('1998-12-01','YYYY-MM-DD') - 90
GROUP BY L_RETURNFLAG, L_LINESTATUS
ORDER BY L_RETURNFLAG, L_LINESTATUS
```

L_RETURNFLAG	L_LINESTATUS	SUM_QTY	SUM_BASE_PRICE
SUM_DISC_PRICE	SUM_CHARGE	AVG_QTY	COUNT_ORDER
A	F	3773034.00	531932989.68
5053976845.78	5256336547.68	25.51	
35964.01	0.05	147907.00	
N	F	100245.00	141459686.10
134380852.77	139710306.87	25.62	
36160.45	0.05	3912.00	
N	O	7464940.00	10518546073.98
9992072944.46	10392414192.06	25.54	
35990.13	0.05	292262.00	
R	F	3779140.00	5328886172.99
5062370635.93	5265431221.82	25.55	
36025.46	0.05	147920.00	

4 rows processed.
Statement Processed in 0.06 seconds.

Ended Executing this Query at Wed Feb 10 16:38:46 1999

Query Started at 918693526.03
Query Ended at 918693526.09
Query Processed in 0.06 seconds.

-- Query 2 (Variant A)

Began Executing this Query at Wed Feb 10 16:38:46 1999

```
select * from (
SELECT
S_ACCTBAL,
S_NAME,
N_NAME,
P_PARTKEY,
P_MFGR,
S_ADDRESS,
S_PHONE,
S_COMMENT
FROM PARTS, 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 (P_PARTKEY, PS_SUPPLYCOST) IN
(SELECT
PS_PARTKEY,
MIN(PS_SUPPLYCOST)
FROM PARTSUPP, SUPPLIER, NATION, REGION
WHERE S_SUPPKEY = PS_SUPPKEY
AND S_NATIONKEY = N_NATIONKEY
AND N_REGIONKEY = R_REGIONKEY
AND R_NAME = 'EUROPE'
GROUP BY PS_PARTKEY)
ORDER BY S_ACCTBAL DESC, N_NAME, S_NAME, P_PARTKEY)
where rownum <=100
```

S_ACCTBAL	S_NAME	N_NAME
P_PARTKEY	P_MFGR	S_PHONE
9828.21	Supplier#000000647	UNITED KINGDOM
13120.00	Manufacturer#5	
jB16PyPyB7b152jMj5Pw3mS		33-258-202-4782
z1QhSiMj11Bm7CO1Lwh6Q10B1R2Mg4CLn		
Lh1P0wiMzy72h1kP715in2y6RS6N1301z51nSR15gOg5S26hPCCQN2L		
9508.37	Supplier#000000070	FRANCE
3563.00	Manufacturer#1	
M5C616R5h5S1MR3zzmLkSw24j2		16-821-608-1166
m7z0CPShmBkhlChBAi3LkQ2CLW		
mhl6QP362RPS3044CB2y4lyhOhj1Bin0CL7yhxmhS4hBM07kQ1yyjOjz3C		
9508.37	Supplier#000000070	FRANCE
17268.00	Manufacturer#4	

M5C616R5h5S1MR3zzmLkSw24j2	16-821-608-1166
m7z0CPShmBkhlChBAi3LkQ2CLW	
mhl6QP362RPS3044CB2y4lyhOhj1Bin0CL7yhxmhS4hBM07kQ1yyjOjz3C	
9453.01	Supplier#000000802
10021.00	Manufacturer#5
5yARQNSLNRAL01BnkNQCIk3S01yClk7nmRHA2h0	29-342-882-6463
65y3RQ2i00P6Nz7mS hC PwLy7L1jQy60163xO3iBCz52Rm1zm0MziCMLij2n6wky51mBowx Qh52iz	
QB1545Amxyj	
9453.01	Supplier#000000802
13275.00	Manufacturer#4
5yARQNSLNRAL01BnkNQCIk3S01yClk7nmRHA2h0	29-342-882-6463
65y3RQ2i00P6Nz7mS hC PwLy7L1jQy60163xO3iBCz52Rm1zm0MziCMLij2n6wky51mBowx Qh52iz	
QB1545Amxyj	
9192.10	Supplier#000000115
13325.00	Manufacturer#1
h0m31z1SPMw2B0ny7LNYNckjRRn7iyMLBLA	33-597-248-1220
1QzQjhSyx	
1xm21gz2Ry7075RL3MS5z36x56hxmR0wLN01Lxml164LzCmMAlzOAjnkz7i4wj01CON11C51M7nCMx66SB	
RAQA	
9032.15	Supplier#000000959
4958.00	Manufacturer#4
205LNCzxMcnQ5gnz4n S3ynP6Mhnx	17-108-642-3106
Px z7kOx5617jQz NwBBQhky yM7kLgxRQw5z6w	426Bm551C6
OkQ7hQPLixjM7y47BNP16CR10kj3541gxh	
8702.02	Supplier#000000333
11810.00	Manufacturer#3
5iwkGn5n2BN150mQk2602h0N6NzxPyiPN51nj	32-508-202-6136
SgimAjmn3wL7R1xmh3LCwOPnhjy1 7xxzxAN 4ACx43y65NwQ7P	
8615.50	Supplier#000000812
10551.00	Manufacturer#2
h412M200 ky1g2m1B0mxjzj0hA2h6nksNPN	16-585-724-6633
5710NayR0RP2jOh54C6B2201SL	
8615.50	Supplier#000000812
13811.00	Manufacturer#4
h412M200 ky1g2m1B0mxjzj0hA2h6nksNPN	16-585-724-6633
5710NayR0RP2jOh54C6B2201SL	
8488.53	Supplier#000000367
6854.00	Manufacturer#4
nkMQZQzgh0wA 3x Sn2S7N5gmSOj xwC COSn6	32-458-198-9557
35C2RR0P C Nlgi2N SxAj0hQkn7Kp5z4w5xSwgMxj6k4MRmh0S2Qm7R3z4jB OOQBML	
8430.52	Supplier#000000646
11384.00	Manufacturer#3
615jP6S y B0 32111	16-601-220-5489
k1w4NSNBKxy5kywzwyx0FMM21xiMOhxR423AkkM Q7CNwRzQS23Nz22 mnm6P377Q3Mj7n	
56BLm6Lxw1lh k5NM	
8271.39	Supplier#000000146
4637.00	Manufacturer#5
wh yPSK6hNB1B4133iQ0wS0 RhBhQ4zQ3lz	32-792-619-3155
jJwgljR063 n7OM2MP0hg3L1mlwBMLm1S4Cgyn LA5PwC2POAS6g3C5mkOj072NPig731m	
8096.98	Supplier#000000574
323.00	Manufacturer#4
hCOj4Cgpx43xx jgP4QkL7gLN65	32-866-246-8752
OhNjG6S1B56315B3k5SCBzwQyLk76z1j4Ow2QB2wACKxh3S0RCy6nARzSQR2010k0BCPhOg6yQm	
7392.78	Supplier#000000170
7655.00	Manufacturer#2
PCxjzjNQiLNXg1w0SiMmQ	33-803-340-5398
M1661xzg54iC3k7OPLQ13Cimbghz2BCLQk g5Ag12QSB1hglANnw4MR MBS 72A	
7205.20	Supplier#000000477
10956.00	Manufacturer#5
Mimj6403h zmAzAgg Bjy050 2z	17-180-144-7991
yRlyR SnMxmhp1AmBw S02AxQ6yOhBRL0wzmlxz00A2Sx075kjlaknn7z2	
00S7hy0BiknwOQm6Pmz3gLGj2Z7	
6820.35	Supplier#000000007
13217.00	Manufacturer#5
z45m2jBRz1511LNz4	33-990-965-2201
1PhngjmiSQ10RzRACFP014S70xSLQPSBM16072SkMLCgm40OMjARLNQk3g1P3BB32AgBM1462B0CP7Rh24	
6721.70	Supplier#000000954
4191.00	Manufacturer#3
OM7xnNxNnkqQ mzh2g3RQmg1g	16-537-341-8517
5ni3yCkms5ymx0kCg74zhLA B516S11w152Ak1Byx1N1	
NgghAKkmNz1jASj4mxzxznOySg7hAyM3MRNBj	
6329.90	Supplier#000000996
10735.00	Manufacturer#2
k6135gA3zPwN17L3R145mlnACjngOQBB300iYA	17-447-811-3282
PBO7w1jQMm1h3AAA 1NQAl0kkiJnkRNgQ0 mhlz6QS0gC51Pl ykmzNR2001N506ARS0z3j	
6173.87	Supplier#000000408
18139.00	Manufacturer#1
Cn16 zR5C41h104POx5h05 mg53CQ2Sw4SAM2M2x	32-858-724-2950
10SxMowhJON3kzhzQ124gNnyw7B4nL7m14L511SR	
5364.99	Supplier#000000785
13784.00	Manufacturer#4
71OnPzQkC2P1hRNRGjyQ4n1	32-297-653-2203
ki1PQ3ik7R ykAhRx43Rw70L10k 7AMi3AjRw71klwxwyiL6S201COyS4Q46m5M167mjMwCm0w	
5069.27	Supplier#000000328
16327.00	Manufacturer#1
504033xSgml	17-231-513-5721
OMk3ALAPNmj61BLMAS7MinCAS 4xLj51iy2kl1x3nP126gAxPgANmk6zS16 3A7m	
11180w1C6xL54hBR1PM	
4941.88	Supplier#000000321
7320.00	Manufacturer#5
hyLQ mg42S2kAM1j M3BwMSjS	29-573-279-1406
y2644kMhOkPCm5P5y7Lmz7OR6mgSmBN631RggmC	
4672.25	Supplier#000000239
12238.00	Manufacturer#1
y4ymj7B5BN1nMSkwPPggaL	32-396-654-6826
Py3RA2gykmSCmj0z3ii7Rxzhz6OyR Rxs C3S23LPQ	
4586.49	Supplier#000000680
5679.00	Manufacturer#3
BP1N1w5nPMxRnOAwm	32-522-382-1620
kAoy2SRN01Alim7SyjPzSym3MS0S52168576km0S2k0mPBL1AzL6Ax7CM6iNi4CgCy6B1n7hlhxm1Rng	
4518.31	Supplier#000000149
18344.00	Manufacturer#5
4B QSy5B12	16-660-553-2456
hi3kPhg1L94L1Q27y0Q42wh0Qz3jPiL4NgkM4NnQ1 1lQ1yNNBk1C1Qn1R07 4ki	
4315.15	Supplier#000000509
18972.00	Manufacturer#2
B5 iPRn7L4yM1lgwCnRPMA	16-298-154-3365
ygiPh7ymP7jBznmr2QLLgjmilwik	
3526.53	Supplier#000000553
8036.00	Manufacturer#4
y1LOx2gMw 5iB16AiNL60Q	16-599-552-3755
L3ggSh1RlyxmR4MN17Rw70Q1gn6yO	
3526.53	Supplier#000000553
17018.00	Manufacturer#3

```
ylLOx2gMw 5iBl6AiNL60Q 16-599-552-3755 58117.00 292632.83 1995-02-21
L3ggShlRlyxmR4MN17Rw7OQign6yO 0.00
3294.68 Supplier#000000350 GERMANY 538311.00 279666.00 1995-03-07
4841.00 Manufacturer#4 0.00
x5kRL2z1BPgO BO 2h1i1Oyh 30Rg00Pj 17-113-181-4017 588421.00 275477.12 1995-03-03
BjQznni44OmQ7S16y13zxx2M6mM4M 27yMPL 0.00
2972.26 Supplier#000000016 RUSSIA 416167.00 273765.45 1995-02-22
1015.00 Manufacturer#4 0.00
B7wLkSLRjNS MS1C 32-822-502-4215 97830.00 273227.06 1995-03-04
C7w6S6QzhAPqmMmMnMN1ha011QOA 00m1NmC25wyQ461SA jy03zmRh222MLM00zhmi 0.00
2963.09 Supplier#000000840 ROMANIA 90276.00 272233.92 1995-03-04
3080.00 Manufacturer#2 0.00
lynwiQkNh0 CMRRCK41306M2i1j0jykg6QNgSCAzy 29-781-337-5584
S7NRMx43Rm0jxML6hxLyN75LzxBwB0wjSLx3 S3Cwh52S6i1SOLhQm0 6C1 yzx3jPm6Sjg
5By0BCPwOR32i1lCQgxR0gB43gB
2221.25 Supplier#000000771 ROMANIA
13981.00 Manufacturer#2
LAjCRj13nAMzzhmw0Sx1Mg 29-986-304-9006
jhhK0N7N1hS23iCngC52BBC 0jilM0wByx0LB5R070R2lCx113lQ187xNhBRA0xkn1NxLiA
1381.97 Supplier#000000104 FRANCE
18103.00 Manufacturer#3
i Qn14 1 jiwM C2yxAYll5R4SBQh54N6 16-434-972-6922
MwmBw1g71Pig2Am7nz0Mm5SN170wQLAkn56ji
906.07 Supplier#000000138 ROMANIA
8363.00 Manufacturer#4
1iBxSxLL1Mh3 6LS6P1LPnl1MjCQh22z6n5 29-533-434-6776
nLjQAmCw77R2jRMgz5LSyry1QN 1 4jMM03RAKxOkzRmwQl3Qm5236k72RRPnim0 BkzQnBMM6A
PMm12n
765.69 Supplier#000000799 RUSSIA
11276.00 Manufacturer#2
Am7y1hz47mg NkgQL w By4 32-579-339-1495
MMRPNQ 4166mQPPNniAi1iLOPQ2C4yyBRn1nRlxxkj5Ak45Pw mQk1ROhz66BRQiil gPRQRy 56MyQ
nS1N14R 7M16xhl210S3
727.89 Supplier#000000470 ROMANIA
6213.00 Manufacturer#3
gAySBM2N7 PgwP5kiP4n7Bz0ik0M 29-165-289-1523
zCkPgn 6wNSA3R47glj1Q3hNSLSHP2RALxClinkOy4wCwA1LCiB05yiSC yBAA lii
683.07 Supplier#000000651 RUSSIA
4888.00 Manufacturer#4
ymQ6PByCh41zxBBPLB2wwOhRh47wMQSPL 32-181-426-4490
kx6jhQkwz96RkRgPLPM30BgLiR72611m5AMk0MmMQBQnCih1LxhMgCgRih6MmMx0Pg1RQ7aQn172g50
167.56 Supplier#000000290 FRANCE
2037.00 Manufacturer#1
xm6n30QmBn75QPPh7Py011xlnB4n11Nh MB11 16-675-286-5102
gi6y41Bg5 AkhmCh30Qyxx515xLi6MRjCQ4n2xjML7N0PNSCyBPwS1C
91.39 Supplier#000000949 UNITED KINGDOM
9430.00 Manufacturer#2
zCmkwm43LQ62Q30 4nkNA 33-332-697-2768
Nm5yn23SxgPAMRz6B5CjAj04nkN51A70
-314.06 Supplier#000000510 ROMANIA
17242.00 Manufacturer#4
hwx1kP6nQ0NgP0232nxAwL4QPjRN 164j0iPjC0 29-207-852-3454
5NAyRmwSCQ5hMB6RiN0Qyhjghn10BPjP1N1P725AAOCw
-820.89 Supplier#000000409 GERMANY
2156.00 Manufacturer#5
Nk4z1lg25gNwMw0 2BnM0n1P3k0yA7i2l 17-719-517-9836
CmlnngQywQ4hhx3266yz5Qgj nOm1P1RC3gxmg4kmmNCSnM4PBO4hNQAlxyz2CLQCBP07w2z3
-845.44 Supplier#000000704 ROMANIA
9926.00 Manufacturer#5
zi jjm04A57 mQgLmhALCM3B zL 29-300-896-5991
QSh0PRm6ByyS2wLBNFQPP12CL0zN0Px1LNO1w16y0Rgi j zP5hPCz
745m62hP0R16i75ynBCC4Qy4Qkn1 1
-942.73 Supplier#000000563 GERMANY
5797.00 Manufacturer#1
xSLLy 4jml70C 17-108-537-2691
1gSol3A0jnkAmQLl1mygzm7hhjjAmRxnSSwJxM16Ok61 wm5N42

44 rows processed.
Statement Processed in 5.21 seconds.

Ended Executing this Query at Wed Feb 10 16:38:51 1999

Query Started at 918693526.09
Query Ended at 918693531.30
Query Processed in 5.21 seconds.

-----

-- Query 3 (Original)

Began Executing this Query at Wed Feb 10 16:38:51 1999

select * from (
SELECT
L_ORDERKEY,
SUM(L_EXTENDEDPRI*(1-L_DISCOUNT)) AS REVENUE,
TO_CHAR(O_ORDERDATE, 'YYYY-MM-DD'),
O_SHIPPRIORITY
FROM CUSTOMER, ORDERS, LINEITEM
WHERE C_MKTSEGMENT = 'BUILDING'
AND C_CUSTKEY = O_CUSTKEY
AND L_ORDERKEY = O_ORDERKEY
AND O_ORDERDATE < TO_DATE('1995-03-15','YYYY-MM-DD')
AND L_SHIPDATE > TO_DATE('1995-03-15','YYYY-MM-DD')
GROUP BY L_ORDERKEY, O_ORDERDATE, O_SHIPPRIORITY
ORDER BY REVENUE DESC, O_ORDERDATE)
where rownum <=10

L_ORDERKEY REVENUE TO_CHAR(O_ORDERDATE,'YYYY-MM-DD')
O_SHIPPRIORITY
260930.00 320547.25 1995-03-12
0.00
402497.00 298879.53 1995-02-12
0.00
457859.00 296490.68 1995-01-17
0.00
509889.00 294068.87 1995-02-03
0.00

-----

-- Query 4 (Original)

Began Executing this Query at Wed Feb 10 16:38:52 1999

SELECT
O_ORDERPRIORITY,
COUNT(*) AS ORDER_COUNT
FROM ORDERS
WHERE O_ORDERDATE >= TO_DATE('1993-07-01','YYYY-MM-DD')
AND O_ORDERDATE < ADD_MONTHS(TO_DATE('1993-07-01','YYYY-MM-DD'),3)
AND EXISTS
(SELECT
*
FROM LINEITEM
WHERE L_ORDERKEY = O_ORDERKEY
AND L_COMMITDATE < L_RECEIPTDATE)
GROUP BY O_ORDERPRIORITY
ORDER BY O_ORDERPRIORITY

O_ORDERPRIORITY ORDER_COUNT
1-URGENT 999.00
2-HIGH 1002.00
3-MEDIUM 1021.00
4-NOT SPECIFIED 997.00
5-LOW 1089.00

5 rows processed.
Statement Processed in 0.66 seconds.

Ended Executing this Query at Wed Feb 10 16:38:52 1999

Query Started at 918693532.17
Query Ended at 918693532.83
Query Processed in 0.66 seconds.

-----

-- Query 5 (Original)

Began Executing this Query at Wed Feb 10 16:38:52 1999

SELECT
N_NAME,
SUM(L_EXTENDEDPRI*(1-L_DISCOUNT)) AS REVENUE
FROM CUSTOMER, ORDERS, LINEITEM, SUPPLIER, NATION, REGION
WHERE C_CUSTKEY = O_CUSTKEY
AND O_ORDERKEY = L_ORDERKEY
AND L_SUPPKEY = S_SUPPKEY
AND C_NATIONKEY = S_NATIONKEY
AND S_NATIONKEY = N_NATIONKEY
AND N_REGIONKEY = R_REGIONKEY
AND R_NAME = 'ASIA'
AND O_ORDERDATE >= TO_DATE('1994-01-01','YYYY-MM-DD')
AND O_ORDERDATE < ADD_MONTHS(TO_DATE('1994-01-01','YYYY-MM-DD'),12)
GROUP BY N_NAME
ORDER BY REVENUE DESC

N_NAME REVENUE
CHINA 7349391.47
INDONESIA 6485853.40
INDIA 5505346.82
JAPAN 5388883.59
VIETNAM 4728846.60

5 rows processed.
Statement Processed in 0.05 seconds.

Ended Executing this Query at Wed Feb 10 16:38:52 1999

Query Started at 918693532.83
Query Ended at 918693532.87
Query Processed in 0.05 seconds.

-----

-- Query 6 (Original)
```

Began Executing this Query at Wed Feb 10 16:38:52 1999

```
SELECT
SUM(L_EXTENDEDPRI * L_DISCOUNT) AS REVENUE
FROM LINEITEM
WHERE L_SHIPDATE >= TO_DATE('1994-01-01','YYYY-MM-DD')
AND L_SHIPDATE < ADD_MONTHS(TO_DATE('1994-01-01','YYYY-MM-DD'),12)
AND L_DISCOUNT BETWEEN .06 - 0.01 AND .06 + 0.01
AND L_QUANTITY < 24
```

REVENUE
11450588.04

1 row processed.
Statement Processed in 0.01 seconds.

Ended Executing this Query at Wed Feb 10 16:38:52 1999

Query Started at 918693532.87
Query Ended at 918693532.88
Query Processed in 0.01 seconds.

-- Query 7 (Original)

Began Executing this Query at Wed Feb 10 16:38:52 1999

```
SELECT
SUPP_NATION,
CUST_NATION,
YEAR,
SUM(VOLUME) AS REVENUE
FROM
(SELECT
N1.N_NAME AS SUPP_NATION,
N2.N_NAME AS CUST_NATION,
TO_CHAR(L_SHIPDATE,'YYYY') AS YEAR,
L_EXTENDEDPRI * (1-L_DISCOUNT) AS VOLUME
FROM SUPPLIER, LINEITEM, ORDERS, 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 TO_DATE('1995-01-01','YYYY-MM-DD')
AND TO_DATE('1996-12-31','YYYY-MM-DD')
) SHIPPING
GROUP BY SUPP_NATION, CUST_NATION, YEAR
ORDER BY SUPP_NATION, CUST_NATION, YEAR
```

SUPP_NATION	CUST_NATION	YEAR	REVENUE
FRANCE	GERMANY	1995	4611421.44
FRANCE	GERMANY	1996	4828420.37
GERMANY	FRANCE	1995	6755766.84
GERMANY	FRANCE	1996	5810951.40

4 rows processed.
Statement Processed in 0.11 seconds.

Ended Executing this Query at Wed Feb 10 16:38:52 1999

Query Started at 918693532.88
Query Ended at 918693533.00
Query Processed in 0.11 seconds.

-- Query 8 (Variant B)

Began Executing this Query at Wed Feb 10 16:38:52 1999

```
SELECT
YEAR,
SUM(DECODE(NATION, 'BRAZIL', VOLUME, 0)) / SUM(VOLUME) AS MKT_SHARE
FROM
(SELECT
TO_CHAR(O_ORDERDATE,'YYYY') AS YEAR,
L_EXTENDEDPRI*(1-L_DISCOUNT) AS VOLUME,
N2.N_NAME AS NATION
FROM PARTS, SUPPLIER, LINEITEM, ORDERS, 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 TO_DATE('1995-01-01','YYYY-MM-DD')
AND TO_DATE('1996-12-31','YYYY-MM-DD')
AND P_TYPE = 'ECONOMY ANODIZED STEEL'
) ALL_NATIONS
GROUP BY YEAR
ORDER BY YEAR
```

YEAR	MKT_SHARE
1995	0.05
1996	0.09

2 rows processed.
Statement Processed in 2.53 seconds.

Ended Executing this Query at Wed Feb 10 16:38:55 1999

Query Started at 918693533.00
Query Ended at 918693535.53
Query Processed in 2.53 seconds.

-- Query 9 (Original)

Began Executing this Query at Wed Feb 10 16:38:55 1999

```
SELECT
NATION,
YEAR,
SUM(AMOUNT) AS SUM_PROFIT
FROM
(SELECT
N_NAME AS NATION,
TO_CHAR(O_ORDERDATE,'YYYY') AS YEAR,
L_EXTENDEDPRI * (1-L_DISCOUNT) - PS_SUPPLYCOST * L_QUANTITY AS AMOUNT
FROM PARTS, SUPPLIER, LINEITEM, PARTSUPP, ORDERS, 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%'
) PROFIT
GROUP BY NATION, YEAR
ORDER BY NATION, YEAR DESC
```

NATION	YEAR	SUM_PROFIT
ALGERIA	1998	1946316.01
ALGERIA	1997	2973825.69
ALGERIA	1996	3308881.52
ALGERIA	1995	3092227.30
ALGERIA	1994	3406958.71
ALGERIA	1993	3140744.03
ALGERIA	1992	3330704.41
ARGENTINA	1998	3045410.01
ARGENTINA	1997	4255378.59
ARGENTINA	1996	4651751.94
ARGENTINA	1995	4897797.00
ARGENTINA	1994	4823465.77
ARGENTINA	1993	4499810.71
ARGENTINA	1992	4764593.39
BRAZIL	1998	2932051.36
BRAZIL	1997	3784531.35
BRAZIL	1996	3965665.69
BRAZIL	1995	4063060.86
BRAZIL	1994	4236277.35
BRAZIL	1993	4363461.31
BRAZIL	1992	4684749.23
CANADA	1998	2217064.04
CANADA	1997	2950110.61
CANADA	1996	3184049.97
CANADA	1995	3962540.19
CANADA	1994	3365251.02
CANADA	1993	3617013.37
CANADA	1992	3407955.25
CHINA	1998	3048192.02
CHINA	1997	5001207.69
CHINA	1996	4800958.31
CHINA	1995	5154927.73
CHINA	1994	5882634.53
CHINA	1993	4733364.82
CHINA	1992	5014704.08
EGYPT	1998	1892538.74
EGYPT	1997	3849220.07
EGYPT	1996	3418656.55
EGYPT	1995	3766170.60
EGYPT	1994	3520025.56
EGYPT	1993	4375424.75
EGYPT	1992	4586034.39
ETHIOPIA	1998	1860117.73
ETHIOPIA	1997	3705722.33
ETHIOPIA	1996	3577215.39
ETHIOPIA	1995	3425219.55
ETHIOPIA	1994	3428616.18
ETHIOPIA	1993	3459815.43
ETHIOPIA	1992	3280072.91
FRANCE	1998	1592531.55
FRANCE	1997	2746176.54
FRANCE	1996	2505844.88
FRANCE	1995	2902077.00
FRANCE	1994	2532229.56
FRANCE	1993	2305725.44
FRANCE	1992	2955126.69
GERMANY	1998	3538625.73
GERMANY	1997	4425943.40
GERMANY	1996	4266344.96
GERMANY	1995	3952963.52
GERMANY	1994	4462655.80
GERMANY	1993	4435094.66
GERMANY	1992	4521715.41
INDIA	1998	3378369.34
INDIA	1997	4186477.85
INDIA	1996	5074383.92
INDIA	1995	4487435.38
INDIA	1994	4718312.63
INDIA	1993	4499573.81

INDIA 1992 4712930.33
INDONESIA 1998 2902077.10
INDONESIA 1997 4973644.23
INDONESIA 1996 4977652.49
INDONESIA 1995 5359380.15
INDONESIA 1994 4854637.20
INDONESIA 1993 4213131.42
INDONESIA 1992 4999478.51
IRAN 1998 2415763.10
IRAN 1997 4227175.11
IRAN 1996 4527365.03
IRAN 1995 4139514.72
IRAN 1994 4166316.39
IRAN 1993 3366959.59
IRAN 1992 3599399.70
IRAQ 1998 2596922.63
IRAQ 1997 3707054.11
IRAQ 1996 3726138.38
IRAQ 1995 4350503.89
IRAQ 1994 4131512.79
IRAQ 1993 3787196.42
IRAQ 1992 4043738.13
JAPAN 1998 2265666.94
JAPAN 1997 3988819.28
JAPAN 1996 4319004.53
JAPAN 1995 4262698.64
JAPAN 1994 3545212.62
JAPAN 1993 4051565.97
JAPAN 1992 3692137.45
JORDAN 1998 1978591.74
JORDAN 1997 3315454.29
JORDAN 1996 3236531.98
JORDAN 1995 2778207.99
JORDAN 1994 2420301.07
JORDAN 1993 3272130.93
JORDAN 1992 2649126.09
KENYA 1998 2265677.73
KENYA 1997 3493019.32
KENYA 1996 3346373.30
KENYA 1995 3537360.32
KENYA 1994 2800950.72
KENYA 1993 3477468.30
KENYA 1992 2719618.04
MOROCCO 1998 2549499.93
MOROCCO 1997 3891824.90
MOROCCO 1996 3730777.74
MOROCCO 1995 3469641.13
MOROCCO 1994 3747593.21
MOROCCO 1993 3620742.70
MOROCCO 1992 4303609.25
MOZAMBIQUE 1998 2024719.46
MOZAMBIQUE 1997 3706003.09
MOZAMBIQUE 1996 3376430.93
MOZAMBIQUE 1995 2737631.64
MOZAMBIQUE 1994 3373146.48
MOZAMBIQUE 1993 3608300.37
MOZAMBIQUE 1992 3551263.95
PERU 1998 2142791.97
PERU 1997 4664076.15
PERU 1996 3623628.93
PERU 1995 3908939.79
PERU 1994 3386204.16
PERU 1993 3877048.49
PERU 1992 3768394.25
ROMANIA 1998 1760625.70
ROMANIA 1997 2707685.33
ROMANIA 1996 2553345.48
ROMANIA 1995 2715901.59
ROMANIA 1994 3023644.06
ROMANIA 1993 2873247.32
ROMANIA 1992 2728060.71
RUSSIA 1998 2975973.22
RUSSIA 1997 3785806.47
RUSSIA 1996 4217625.59
RUSSIA 1995 3883445.52
RUSSIA 1994 4395855.01
RUSSIA 1993 3900944.18
RUSSIA 1992 4691358.61
SAUDI ARABIA 1998 2931482.83
SAUDI ARABIA 1997 5498943.16
SAUDI ARABIA 1996 4473723.74
SAUDI ARABIA 1995 5939212.93
SAUDI ARABIA 1994 4527695.71
SAUDI ARABIA 1993 4928702.02
SAUDI ARABIA 1992 5527261.52
UNITED KINGDOM 1998 3198731.37
UNITED KINGDOM 1997 4363882.74
UNITED KINGDOM 1996 4730956.67
UNITED KINGDOM 1995 4842014.55
UNITED KINGDOM 1994 4912706.56
UNITED KINGDOM 1993 4415255.96
UNITED KINGDOM 1992 4375524.23
UNITED STATES 1998 1892045.16
UNITED STATES 1997 3102027.86
UNITED STATES 1996 3334320.26
UNITED STATES 1995 3168244.60
UNITED STATES 1994 3296960.10
UNITED STATES 1993 3558109.05
UNITED STATES 1992 2755129.39
VIETNAM 1998 2906627.03
VIETNAM 1997 4544560.45
VIETNAM 1996 4314259.00
VIETNAM 1995 4365340.86
VIETNAM 1994 3686987.71
VIETNAM 1993 3764237.18
VIETNAM 1992 3420922.00

175 rows processed.
Statement Processed in 3.75 seconds.

Ended Executing this Query at Wed Feb 10 16:38:59 1999

Query Started at 918693535.53
Query Ended at 918693539.28
Query Processed in 3.75 seconds.

-- Query 10 (Original)

Began Executing this Query at Wed Feb 10 16:38:59 1999

```
select * from (
SELECT
  C_CUSTKEY,
  C_NAME,
  SUM(L_EXTENDEDPRI * (1-L_DISCOUNT)) AS REVENUE,
  C_ACCTBAL,
  N_NAME,
  C_ADDRESS,
  C_PHONE,
  C_COMMENT
FROM CUSTOMER, ORDERS, LINEITEM, NATION
WHERE C_CUSTKEY = O_CUSTKEY
AND L_ORDERKEY = O_ORDERKEY
AND O_ORDERDATE >= TO_DATE('1993-10-01', 'YYYY-MM-DD')
AND O_ORDERDATE < ADD_MONTHS(TO_DATE('1993-10-01', 'YYYY-MM-DD'), 3)
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)
where rownum <=20

C_CUSTKEY          C_NAME          REVENUE
C_ACCTBAL          N_NAME
C_ADDRESS          C_PHONE
C_COMMENT
9722.00            Customer#000009722      464618.26
474.04             CANADA
1 Mwzn4NAk6j      13-518-602-8070
5L 500y RSgBAzPxmoS15wk6xxOR7kh2nnPly7L8ng2hOw5B01
RmCM12L24Pkq7PS1zwc11BCn24L615SPK18P26166
12800.00           Customer#000012800      444265.64
1900.84            PERU
57zjB3CQx4P40B2R2MBi2mwhS11M4mn 4 nC6 27-142-205-3552
0hwqLS77RB56Rx4361QON16CxbOPmmyhwgW
5z64wnj1kiC4jL350mM4ly71hNxBl1PjyA4hiN1wzjM7SCxAN244mk2A
1025.00            Customer#000001025      442028.02
3363.46            INDIA
1kiSn154MSROi      18-588-456-4616
0B145z233Rniw0O0064nPBgP16kimO0y74iLh73g1N4m310jQyQzPA50iC3MA75g2Bj162Nw4P
13028.00           Customer#000013028      441692.24
-452.66            UNITED KINGDOM
yP714ORSNN2LA3LSB 33-253-660-2127
xPkmnhL2BkhkNyw4khlxwAymNh11PSjBCNMI5OLkyOhO6CC5nzOQCALzliOk2R66w105hRPO3iSP
3694.00            Customer#000003694      438180.07
2960.44            UNITED KINGDOM
2CCk1mCBOCC        33-421-331-3127
MzLxQxLlL3xMPx1Awg1B5kg61zxkPnk xiAm6PhMMAAQ2nzN3S6zzgP
x70w0lhhPx4QRz1MMY02041Al3mB07jh2jAPON60wg367z
976.00             Customer#000000976      435897.63
7772.85            ROMANIA
QzR 56Px1kgS wANnAz02RS 30n Pm 29-436-660-4732
kzn12776 gwzKzZZzO4yxOAnkR7hR4R4x2SMw1z3x6h nN7OnNLRmM13 kz5SLwilyk1OxiwS4g0wmA5A
4hmgBSWR1Q1
8206.00            Customer#000008206      429905.11
6046.36            ARGENTINA
P yMg30BBBx NMgC03AmzN2 11-571-859-1370
hLi122RPMcL36Oy0kxO71zz2wCR0QQCl7z26h1Q3mM
13532.00           Customer#000013532      427731.80
-924.18            KENYA
61j7MSPBxK2kwwyz620j4SL5S0mRCw13mlRmw 24-525-332-7244
7ih7yRz214z067A1Nfx64nO515k yj6i3jLA5PCL15Q4Q1A31160iM1P iBxCixg6 1hCh2RCnjOzk5R
OnO 10hhC3m4631m5
12745.00           Customer#000012745      422327.69
9691.33            CHINA
SgS1LMC4gB2NM3wh 28-985-189-6174
jl72wJsw0 S6 7L4Cgxw PkyO5N12LL7LBR
2344.00            Customer#000002344      411240.11
5597.22            MOROCCO
O3PC7ikBgw OazPALm2P 426zm3BnBN6Q10 6N 25-593-745-7663
5NBn0wRNngLw2z5kyn1AhL0ASyg6SMhM i2kMOyxARAn10Q5j4CBNARix7AB1MAC
2656.00            Customer#000002656      401185.95
8115.55            ALGERIA
On551AS3Rm5Rxs m 10-667-469-8092
46ABx4jgmi m1BMPClXrRhyPQM4RNS 5yO1L7zSOmk MhPxAXQQ61QnLj
17LymOh1451innzOyB2O1xzmw3gmXOSxiyBN5CSMNgCkLCKmGo
59.00              Customer#00000059      400759.15
3458.60            ARGENTINA
wP6CMYClly01S4CAM1zm 11-355-584-3112
1lg7xBcXxc7SM5AkmmAk00677O1MzA2R7A0Cx0Nj ixj56jL2iNPNKSNQiy55m6ki3OgnhM47mSR7B
7069.00           Customer#000007069      396217.52
8198.94            INDONESIA
55Cw7ChL4Bi5ONn2A4m2i2n4nSNQMJml 19-644-744-1798
6jNS624175z1xN1i41x05zyPykPS1xnl180NhkgOAKSx7P
6553.00            Customer#000006553      385863.59
8985.90            MOZAMBIQUE
R3LnnxONBjCLCOMRkxy7 26-166-724-4677
S7CKNLwA3kh006j71lwAlC25Bw6AMQ6i 6C0OSS607ARNNNy600gh 3642mRxyiAgy5yk
3nPO4473wKng5R6g2041z32mWzm7M1A11CC
3095.00            Customer#000003095      384246.11
8829.21            IRAQ
SlgMcNBLwzi mCgB664 jl00L11ShnliPMgCgR5 21-847-218-8188
3LSx7PxS A4A5C13gAy3mq4Qj2xQlyx7xM1kA64AM7zmMnzOrh3Clh
MO3n6MymiljAMg65hOMB48n44kO w0lin7
3391.00            Customer#000003391      382541.78
7742.35            CANADA
m3 CORmQNLzkShymLS iMkCimRS120 NB 13-592-494-2668
ynM1mhMBA5ikC1nCgh1mAhQ0 675S3y2R33yjkNPPQOS
13678.00           Customer#000013678      376280.56
9030.40            MOROCCO
```

```
BMk77lQm1wNA0LghAkg3hCwN14          25-306-951-3937
mOS55RASx1wP136nQ5xBLznLhgwkQ6PO6imNxQ7kR0x71P0SzByMzh
6062.00          Customer#000006062          374512.65
1370.35          CANADA
n5zzil60zyxAlkzx7xlnihigPzR  OBkR  znMOMh  13-756-700-4918
4zAm4wNB 114QRPGpPz2wM541x043hmlJ4O3LBkALCP16hj2RQB01OMNly7ww1QP7w5i SSn0jNnAR
yQmmz1hi5j3
554.00          Customer#000000554          373004.47
8395.57          BRAZIL
jC5zhQky4zQB27lB5Sm AqhQ  Px0          12-938-503-7317
0nxCl3 xSmiLQO 1M
2n0NCiRlNmMxP25j26x2igLhN0xjgMgmwWy7OkjzCACOG0z2LAjOm0RPRmOPiCAAQwLlQSG 1yS3
gLCMI1M2BzjnSjPl3nwAkk
13126.00          Customer#000013126          371722.00
6172.91          INDIA
xPAS4MnPh40i5Q2h4NQ61zz4RkyAwANA          18-288-190-4145
nniMkAN6C0ClQ0mMmPz271iz4hk6L2MLwPxh42N110R2hRwxzxLwMkxO4MAyz7RCj43NxLwQ3m6P27yAj
```

20 rows processed.
Statement Processed in 1.20 seconds.

Ended Executing this Query at Wed Feb 10 16:39:00 1999

Query Started at 918693539.28
Query Ended at 918693540.48
Query Processed in 1.20 seconds.

-- Query 11 (Original)

Began Executing this Query at Wed Feb 10 16:39:00 1999

```
SELECT
PS_PARTKEY,
SUM(PS_SUPPLYCOST * PS_AVAILQTY) 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) * 0.0010000000
FROM PARTSUPP, SUPPLIER, NATION
WHERE PS_SUPPKEY = S_SUPPKEY
AND S_NATIONKEY = N_NATIONKEY
AND N_NAME = 'GERMANY' )
ORDER BY VALUE DESC
```

PS_PARTKEY	VALUE
12098.00	16227681.21
5134.00	15709338.52
13334.00	15023662.41
17052.00	14351644.20
3452.00	14070870.14
12552.00	13332469.18
1084.00	13170428.29
5797.00	13038622.72
12633.00	12892561.61
403.00	12856217.34
1833.00	12024581.72
2084.00	11502875.36
17349.00	11354213.05
18427.00	11282385.24
2860.00	11262529.95
17852.00	10934711.93
9871.00	10889253.68
12231.00	10841131.39
6366.00	10759786.81
12146.00	10257362.66
5043.00	10226395.88
12969.00	10125777.93

22 rows processed.
Statement Processed in 0.62 seconds.

Ended Executing this Query at Wed Feb 10 16:39:01 1999

Query Started at 918693540.48
Query Ended at 918693541.10
Query Processed in 0.62 seconds.

-- Query 12 (Variant B)

Began Executing this Query at Wed Feb 10 16:39:01 1999

```
SELECT
L_SHIPMODE,
SUM(DECODE(O_ORDERPRIORITY,'1-URGENT',1,'2-HIGH',1,0)) AS HIGH_LINE_COUNT,
SUM(DECODE(O_ORDERPRIORITY,'1-URGENT',0,'2-HIGH',0,1)) AS LOW_LINE_COUNT
FROM ORDERS, 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 >= TO_DATE('1994-01-01','YYYY-MM-DD')
AND L_RECEIPTDATE < ADD_MONTHS(TO_DATE('1994-01-01','YYYY-MM-DD'),12)
GROUP BY L_SHIPMODE
ORDER BY L_SHIPMODE

L_SHIPMODE HIGH_LINE_COUNT          LOW_LINE_COUNT
```

MAIL	654.00	950.00
SHIP	684.00	1004.00

2 rows processed.
Statement Processed in 0.79 seconds.

Ended Executing this Query at Wed Feb 10 16:39:01 1999

Query Started at 918693541.10
Query Ended at 918693541.89
Query Processed in 0.79 seconds.

-- Query 13 (Original)

Began Executing this Query at Wed Feb 10 16:39:01 1999

```
SELECT
YEAR,
SUM(REVENUE) AS REVENUE
FROM
(SELECT
TO_CHAR(O_ORDERDATE,'YYYY') AS YEAR,
L_EXTENDEDPRI * (1-L_DISCOUNT) AS REVENUE
FROM LINEITEM, ORDERS
WHERE O_ORDERKEY = L_ORDERKEY
AND O_CLERK = 'Clerk#0000000088'
AND L_RETURNFLAG = 'R'
) PERFORMANCE
GROUP BY YEAR
ORDER BY YEAR
```

YEAR	REVENUE
1992	1262855.73
1993	964121.03
1994	1750395.29
1995	198820.30

4 rows processed.
Statement Processed in 0.38 seconds.

Ended Executing this Query at Wed Feb 10 16:39:02 1999

Query Started at 918693541.89
Query Ended at 918693542.27
Query Processed in 0.38 seconds.

-- Query 14 (Original)

Began Executing this Query at Wed Feb 10 16:39:02 1999

```
SELECT
100.00 * SUM(CASE WHEN P_TYPE LIKE 'PROMO%'
THEN L_EXTENDEDPRI*(1-L_DISCOUNT)
ELSE 0
END) /SUM(L_EXTENDEDPRI*(1-L_DISCOUNT)) AS PROMO_REVENUE
FROM LINEITEM, PARTS
WHERE
L_PARTKEY = P_PARTKEY AND
L_SHIPDATE >= TO_DATE('1995-09-01','YYYY-MM-DD') AND
L_SHIPDATE < ADD_MONTHS(TO_DATE('1995-09-01','YYYY-MM-DD'), 1)

PROMO_REVENUE
16.73
```

1 row processed.
Statement Processed in 0.03 seconds.

Ended Executing this Query at Wed Feb 10 16:39:02 1999

Query Started at 918693542.27
Query Ended at 918693542.30
Query Processed in 0.03 seconds.

-- Query 15 (Variant B)

Began Executing this Query at Wed Feb 10 16:39:02 1999

```
CREATE TABLE REVENUE0 (SUPPLIER_NO INTEGER, TOTAL_REVENUE NUMBER(20, 2))
Statement Processed in 0.04 seconds.

INSERT INTO REVENUE0
SELECT
L_SUPPKEY,
SUM(L_EXTENDEDPRI*(1-L_DISCOUNT))
FROM LINEITEM
WHERE L_SHIPDATE >= TO_DATE('1996-01-01','YYYY-MM-DD')
AND L_SHIPDATE < ADD_MONTHS(TO_DATE('1996-01-01','YYYY-MM-DD'),3)
GROUP BY L_SUPPKEY
Statement Processed in 1.63 seconds.

COMMIT
Statement Processed in 0.01 seconds.
```

```

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

```

```

S_SUPPKY      S_NAME      S_PHONE      TOTAL_REVENUE
S_ADDRESS
389.00      Supplier#000000389
PB1Lx0xx6LMz3h7Rx63m6j3QmMx      34-885-883-5717 1418538.21

```

1 row processed.
Statement Processed in 1.12 seconds.

DROP TABLE REVENUE0
Statement Processed in 0.05 seconds.

Ended Executing this Query at Wed Feb 10 16:39:05 1999

Query Started at 918693542.30
Query Ended at 918693545.13
Query Processed in 2.84 seconds.

-- Query 16 (Original)

Began Executing this Query at Wed Feb 10 16:39:05 1999

```

SELECT
P_BRAND,
P_TYPE,
P_SIZE,
COUNT(DISTINCT PS_SUPPKY) AS SUPPLIER_CNT
FROM PARTSUPP, PARTS
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_SUPPKY NOT IN
(SELECT
S_SUPPKY
FROM SUPPLIER
WHERE S_COMMENT LIKE '%Better Business Bureau%Complaints%')
GROUP BY P_BRAND, P_TYPE, P_SIZE
ORDER BY SUPPLIER_CNT DESC, P_BRAND, P_TYPE, P_SIZE

```

P_BRAND	P_TYPE	P_SIZE	SUPPLIER_CNT
Brand#14	SMALL ANODIZED NICKEL	45.00	12.00
Brand#22	SMALL BURNISHED BRASS	19.00	12.00
Brand#25	PROMO POLISHED COPPER	14.00	12.00
Brand#35	LARGE ANODIZED STEEL	45.00	12.00
Brand#35	PROMO BRUSHED COPPER	9.00	12.00
Brand#51	ECONOMY ANODIZED STEEL	9.00	12.00
Brand#53	LARGE BRUSHED NICKEL	45.00	12.00
Brand#11	ECONOMY POLISHED COPPER	14.00	8.00
Brand#11	LARGE PLATED STEEL	23.00	8.00
Brand#11	PROMO POLISHED STEEL	23.00	8.00
Brand#11	STANDARD ANODIZED COPPER	9.00	8.00
Brand#12	ECONOMY BURNISHED BRASS	9.00	8.00
Brand#12	LARGE ANODIZED BRASS	14.00	8.00
Brand#12	SMALL ANODIZED TIN	23.00	8.00
Brand#12	SMALL BRUSHED NICKEL	23.00	8.00
Brand#12	STANDARD ANODIZED BRASS	3.00	8.00
Brand#12	STANDARD BURNISHED TIN	23.00	8.00
Brand#13	ECONOMY POLISHED BRASS	9.00	8.00
Brand#13	LARGE BURNISHED COPPER	45.00	8.00
Brand#13	MEDIUM ANODIZED STEEL	23.00	8.00
Brand#13	MEDIUM PLATED NICKEL	3.00	8.00
Brand#13	PROMO BURNISHED BRASS	9.00	8.00
Brand#13	PROMO POLISHED BRASS	3.00	8.00
Brand#13	PROMO POLISHED TIN	36.00	8.00
Brand#13	SMALL BURNISHED STEEL	23.00	8.00
Brand#13	STANDARD BRUSHED STEEL	9.00	8.00
Brand#14	ECONOMY BRUSHED TIN	3.00	8.00
Brand#14	ECONOMY BURNISHED TIN	23.00	8.00
Brand#14	PROMO BRUSHED STEEL	9.00	8.00
Brand#14	PROMO PLATED TIN	45.00	8.00
Brand#15	ECONOMY PLATED TIN	9.00	8.00
Brand#15	STANDARD BRUSHED COPPER	14.00	8.00
Brand#15	STANDARD PLATED TIN	3.00	8.00
Brand#21	ECONOMY POLISHED TIN	3.00	8.00
Brand#21	PROMO POLISHED COPPER	9.00	8.00
Brand#21	PROMO POLISHED TIN	49.00	8.00
Brand#21	STANDARD PLATED BRASS	49.00	8.00
Brand#21	STANDARD PLATED NICKEL	49.00	8.00
Brand#22	ECONOMY ANODIZED TIN	49.00	8.00
Brand#22	ECONOMY BRUSHED BRASS	14.00	8.00
Brand#22	LARGE BURNISHED TIN	36.00	8.00
Brand#22	MEDIUM ANODIZED STEEL	36.00	8.00
Brand#22	MEDIUM PLATED STEEL	9.00	8.00
Brand#22	PROMO POLISHED NICKEL	9.00	8.00
Brand#22	SMALL ANODIZED STEEL	19.00	8.00
Brand#22	STANDARD ANODIZED COPPER	23.00	8.00
Brand#23	ECONOMY BRUSHED NICKEL	23.00	8.00
Brand#23	LARGE ANODIZED BRASS	9.00	8.00
Brand#23	LARGE ANODIZED STEEL	23.00	8.00
Brand#23	SMALL BRUSHED COPPER	23.00	8.00
Brand#23	STANDARD BRUSHED TIN	3.00	8.00
Brand#23	STANDARD BURNISHED NICKEL	49.00	8.00
Brand#23	STANDARD PLATED NICKEL	36.00	8.00
Brand#24	ECONOMY ANODIZED BRASS	19.00	8.00

Brand#24	ECONOMY POLISHED BRASS	36.00	8.00
Brand#24	LARGE BURNISHED STEEL	14.00	8.00
Brand#24	MEDIUM PLATED NICKEL	36.00	8.00
Brand#25	ECONOMY BRUSHED STEEL	49.00	8.00
Brand#25	MEDIUM BURNISHED TIN	3.00	8.00
Brand#25	PROMO ANODIZED TIN	36.00	8.00
Brand#25	PROMO PLATED NICKEL	3.00	8.00
Brand#25	SMALL BURNISHED BRASS	3.00	8.00
Brand#31	LARGE ANODIZED BRASS	3.00	8.00
Brand#31	SMALL ANODIZED COPPER	3.00	8.00
Brand#31	SMALL ANODIZED NICKEL	9.00	8.00
Brand#31	SMALL ANODIZED STEEL	14.00	8.00
Brand#32	MEDIUM ANODIZED STEEL	49.00	8.00
Brand#32	MEDIUM BURNISHED COPPER	19.00	8.00
Brand#32	SMALL BURNISHED STEEL	23.00	8.00
Brand#32	STANDARD BURNISHED STEEL	45.00	8.00
Brand#34	ECONOMY ANODIZED NICKEL	49.00	8.00
Brand#34	LARGE BURNISHED TIN	49.00	8.00
Brand#34	PROMO ANODIZED TIN	3.00	8.00
Brand#34	SMALL BRUSHED TIN	3.00	8.00
Brand#34	STANDARD BURNISHED TIN	23.00	8.00
Brand#35	MEDIUM BRUSHED STEEL	45.00	8.00
Brand#35	PROMO BURNISHED STEEL	14.00	8.00
Brand#35	SMALL BURNISHED STEEL	23.00	8.00
Brand#35	SMALL POLISHED COPPER	14.00	8.00
Brand#35	STANDARD PLATED COPPER	9.00	8.00
Brand#41	ECONOMY BRUSHED BRASS	23.00	8.00
Brand#41	LARGE BURNISHED STEEL	23.00	8.00
Brand#41	PROMO BURNISHED TIN	14.00	8.00
Brand#41	PROMO PLATED STEEL	36.00	8.00
Brand#41	PROMO POLISHED TIN	19.00	8.00
Brand#41	SMALL BURNISHED COPPER	23.00	8.00
Brand#42	LARGE POLISHED TIN	14.00	8.00
Brand#42	MEDIUM ANODIZED TIN	49.00	8.00
Brand#42	MEDIUM BRUSHED TIN	14.00	8.00
Brand#42	MEDIUM BURNISHED NICKEL	23.00	8.00
Brand#42	MEDIUM PLATED COPPER	45.00	8.00
Brand#42	MEDIUM PLATED TIN	45.00	8.00
Brand#42	SMALL PLATED COPPER	36.00	8.00
Brand#43	ECONOMY BRUSHED STEEL	45.00	8.00
Brand#43	LARGE BRUSHED COPPER	19.00	8.00
Brand#43	PROMO BRUSHED BRASS	36.00	8.00
Brand#43	SMALL BURNISHED TIN	45.00	8.00
Brand#43	SMALL PLATED COPPER	45.00	8.00
Brand#44	PROMO POLISHED TIN	23.00	8.00
Brand#44	SMALL POLISHED NICKEL	14.00	8.00
Brand#44	SMALL POLISHED TIN	45.00	8.00
Brand#44	STANDARD BURNISHED COPPER	3.00	8.00
Brand#51	LARGE ANODIZED BRASS	19.00	8.00
Brand#51	LARGE POLISHED COPPER	23.00	8.00
Brand#51	MEDIUM ANODIZED TIN	9.00	8.00
Brand#51	MEDIUM ANODIZED TIN	14.00	8.00
Brand#51	MEDIUM BURNISHED NICKEL	23.00	8.00
Brand#51	SMALL ANODIZED COPPER	45.00	8.00
Brand#51	SMALL ANODIZED COPPER	49.00	8.00
Brand#51	SMALL BRUSHED COPPER	45.00	8.00
Brand#51	SMALL BRUSHED TIN	36.00	8.00
Brand#51	STANDARD POLISHED TIN	3.00	8.00
Brand#52	ECONOMY ANODIZED STEEL	3.00	8.00
Brand#52	ECONOMY PLATED TIN	19.00	8.00
Brand#52	LARGE PLATED TIN	3.00	8.00
Brand#52	MEDIUM ANODIZED TIN	19.00	8.00
Brand#52	MEDIUM BURNISHED COPPER	3.00	8.00
Brand#52	PROMO POLISHED BRASS	23.00	8.00
Brand#52	SMALL PLATED COPPER	36.00	8.00
Brand#52	SMALL POLISHED NICKEL	9.00	8.00
Brand#52	STANDARD POLISHED NICKEL	45.00	8.00
Brand#53	ECONOMY POLISHED STEEL	45.00	8.00
Brand#53	LARGE POLISHED NICKEL	3.00	8.00
Brand#53	SMALL BRUSHED COPPER	14.00	8.00
Brand#53	STANDARD PLATED STEEL	45.00	8.00
Brand#54	ECONOMY POLISHED BRASS	49.00	8.00
Brand#54	ECONOMY POLISHED TIN	23.00	8.00
Brand#54	MEDIUM BRUSHED STEEL	9.00	8.00
Brand#54	SMALL BURNISHED NICKEL	14.00	8.00
Brand#54	SMALL PLATED TIN	14.00	8.00
Brand#54	STANDARD BURNISHED STEEL	14.00	8.00
Brand#54	STANDARD PLATED BRASS	23.00	8.00
Brand#55	MEDIUM BURNISHED TIN	36.00	8.00
Brand#55	PROMO ANODIZED BRASS	14.00	8.00
Brand#55	STANDARD BURNISHED COPPER	45.00	8.00
Brand#15	STANDARD PLATED TIN	36.00	7.00
Brand#21	SMALL POLISHED STEEL	3.00	7.00
Brand#23	SMALL POLISHED BRASS	49.00	7.00
Brand#34	MEDIUM BURNISHED NICKEL	3.00	7.00
Brand#42	STANDARD PLATED COPPER	19.00	7.00
Brand#51	LARGE POLISHED NICKEL	14.00	7.00
Brand#54	LARGE ANODIZED NICKEL	49.00	7.00
Brand#11	ECONOMY ANODIZED BRASS	19.00	4.00
Brand#11	ECONOMY ANODIZED BRASS	45.00	4.00
Brand#11	ECONOMY ANODIZED NICKEL	36.00	4.00
Brand#11	ECONOMY BRUSHED COPPER	3.00	4.00
Brand#11	ECONOMY BRUSHED COPPER	9.00	4.00
Brand#11	ECONOMY BRUSHED STEEL	9.00	4.00
Brand#11	ECONOMY BRUSHED STEEL	36.00	4.00
Brand#11	ECONOMY BURNISHED BRASS	36.00	4.00
Brand#11	ECONOMY BURNISHED COPPER	9.00	4.00
Brand#11	ECONOMY BURNISHED COPPER	49.00	4.00
Brand#11	ECONOMY BURNISHED NICKEL	14.00	4.00
Brand#11	ECONOMY BURNISHED NICKEL	49.00	4.00
Brand#11	ECONOMY PLATED COPPER	19.00	4.00
Brand#11	ECONOMY PLATED NICKEL	45.00	4.00
Brand#11	ECONOMY PLATED TIN	9.00	4.00
Brand#11	ECONOMY POLISHED BRASS	3.00	4.00
Brand#11	ECONOMY POLISHED COPPER	3.00	4.00
Brand#11	ECONOMY POLISHED COPPER	45.00	4.00
Brand#11	ECONOMY POLISHED NICKEL	36.00	4.00
Brand#11	ECONOMY POLISHED STEEL	23.00	4.00
Brand#11	ECONOMY POLISHED TIN	14.00	4.00
Brand#11	LARGE ANODIZED COPPER	23.00	4.00
Brand#11	LARGE ANODIZED NICKEL	9.00	4.00

Brand#11	LARGE ANODIZED STEEL	9.00	4.00
Brand#11	LARGE BRUSHED STEEL	19.00	4.00
Brand#11	LARGE BRUSHED TIN	3.00	4.00
Brand#11	LARGE BRUSHED TIN	14.00	4.00
Brand#11	LARGE BURNISHED COPPER	9.00	4.00
Brand#11	LARGE BURNISHED COPPER	19.00	4.00
Brand#11	LARGE BURNISHED STEEL	23.00	4.00
Brand#11	LARGE BURNISHED TIN	9.00	4.00
Brand#11	LARGE PLATED COPPER	23.00	4.00
Brand#11	LARGE PLATED TIN	9.00	4.00
Brand#11	LARGE PLATED TIN	14.00	4.00
Brand#11	LARGE PLATED TIN	23.00	4.00
Brand#11	LARGE POLISHED NICKEL	49.00	4.00
Brand#11	MEDIUM ANODIZED BRASS	45.00	4.00
Brand#11	MEDIUM ANODIZED TIN	14.00	4.00
Brand#11	MEDIUM BRUSHED BRASS	14.00	4.00
Brand#11	MEDIUM BRUSHED BRASS	45.00	4.00
Brand#11	MEDIUM BRUSHED NICKEL	14.00	4.00
Brand#11	MEDIUM BRUSHED NICKEL	36.00	4.00
Brand#11	MEDIUM BRUSHED STEEL	19.00	4.00
Brand#11	MEDIUM BURNISHED COPPER	9.00	4.00
Brand#11	MEDIUM BURNISHED TIN	36.00	4.00
Brand#11	MEDIUM PLATED BRASS	3.00	4.00
Brand#11	MEDIUM PLATED TIN	19.00	4.00
Brand#11	PROMO ANODIZED BRASS	3.00	4.00
Brand#11	PROMO ANODIZED BRASS	19.00	4.00
Brand#11	PROMO ANODIZED BRASS	45.00	4.00
Brand#11	PROMO ANODIZED BRASS	49.00	4.00
Brand#11	PROMO ANODIZED STEEL	23.00	4.00
Brand#11	PROMO ANODIZED TIN	45.00	4.00
Brand#11	PROMO BRUSHED BRASS	23.00	4.00
Brand#11	PROMO BRUSHED STEEL	3.00	4.00
Brand#11	PROMO BURNISHED BRASS	23.00	4.00
Brand#11	PROMO BURNISHED BRASS	36.00	4.00
Brand#11	PROMO BURNISHED BRASS	49.00	4.00

(rows deleted)

2762 rows processed.
Statement Processed in 1.16 seconds.

Ended Executing this Query at Wed Feb 10 16:39:06 1999

Query Started at 918693545.13
Query Ended at 918693546.30
Query Processed in 1.16 seconds.

-- Query 17 (Variant B)

Began Executing this Query at Wed Feb 10 16:39:06 1999

```

SELECT
SUM(L_EXTENDEDPRICE) / 7.0 AS AVG_YEARLY
FROM LINEITEM,
(SELECT
P_PARTKEY,
0.2 * AVG(L_QUANTITY) AS AVGGQTY
FROM PARTS, LINEITEM
WHERE P_PARTKEY = L_PARTKEY
AND P_BRAND = 'Brand#23'
AND P_CONTAINER = 'MED BOX'
GROUP BY P_PARTKEY) AVG_QUANTITY
WHERE P_PARTKEY = L_PARTKEY
AND L_QUANTITY < AVGGQTY

AVG_YEARLY
24436.88

```

1 row processed.
Statement Processed in 0.46 seconds.

Ended Executing this Query at Wed Feb 10 16:39:06 1999

Query Started at 918693546.30
Query Ended at 918693546.76
Query Processed in 0.46 seconds

SQL statements processed: 21
Queries processed: 17

Appendix D. Seed and Query Substitution Parameters

```
-- using 1259440881 as a seed to the RNG

Substitution parameters

-- Query 1 (Original)
92

-- Query 4 (Original)
1995-08-01

-- Query 15 (Variant B)
1995-08-01

-- Query 10
1994-02-01

-- Query 11
JORDAN 0.0000001000

-- Query 6
1995-01-01 0.09 24

-- Query 2
27 STEEL AFRICA

-- Query 16
Brand#35 STANDARD PLATED 42, 17 ,13 ,30 ,18, 38, 8, 29

-- Query 14
1995-09-01

-- Query 8
JORDAN MIDDLE EAST PROMO BURNISHED STEEL

-- Query 12
TRUCK SHIP 1993-01-01

-- Query 17
Brand#35 SM JAR

-- Query 3
FURNITURE 1995-03-28

-- Query 5
ASIA 1997-01-01

-- Query 13
Clerk#000000533

-- Query 7
JORDAN RUSSIA

-- Query 9
maroon
```


Appendix E. Implementation - Specific Layer/Driver Code

```
audit.sh
#!/bin/ksh
# -----
# Copyright (c) 1996 Oracle Corp, Redwood Shores, CA
# OPEN SYSTEMS PERFORMANCE DIVISION
# All Rights Reserved
# -----
# FILENAME
# audit.sh
# DESCRIPTION
# Usage: audit.sh [<all|first|second>]
#
# Run 2 power runs (audit)
# -----

#
# Get params
#
FIRST="yes"
SECOND="yes"

if [ "$1" = "first" ]; then
    SECOND="no"
else
    if [ "$1" = "second" ]; then
        FIRST="no"
    fi
fi

#
# Set PDML/PDDL degree
#
export PDML_DEGREE=160
export PDDL_DEGREE=64

if [ "$FIRST" = "yes" ]; then
    #
    # Bounce database before first run
    #
    ./tshut
    sleep 60
    ./tstart

    echo "Starting first run"
    date

    #
    # first run
    #
    audit_stream.sh 1000 first

    echo "Done with first run"
    date

    echo
    echo
fi

if [ "$SECOND" = "yes" ]; then
    #
    # Bounce database before second run
    #
    ./tshut
    sleep 60
    ./tstart

    echo "Starting second run"
    date

    #
    # second run
    #
    audit_stream.sh 1000 second

    echo "Done with second run"
    date
fi

wait
./tshut

audit_stream.sh
#!/bin/ksh
# -----
# Copyright (c) 1996 Oracle Corp, Redwood Shores, CA
# OPEN SYSTEMS PERFORMANCE DIVISION
# All Rights Reserved
# -----
# FILENAME
# audit_stream.sh
# DESCRIPTION
# Usage: audit_stream.sh <scale_factor> <first|second>
#
# Run multiple or single stream audit
# -----
TPCD_HOME=/ora81/Audit
UPD_DIR=${TPCD_HOME}/update
SRC_DIR=${TPCD_HOME}/source
QRY_DIR=${TPCD_HOME}/queries
QGEN_DIR=${TPCD_HOME}/bumpx/dbgen
QGEN=${QGEN_DIR}/qgen
```

```
DSS_QUERY=${TPCD_HOME}/Dqueries
export DSS_QUERY

RUN_ID_FILE=${TPCD_HOME}/r_id

TPCD_LOG=${TPCD_HOME}/log
TPCD_RPT=${TPCD_HOME}/rpt

OUT=${TPCD_HOME}/out
GTIME=${SRC_DIR}/gtime
HID=1

# The defaults
USER="tpcd/tpcd"
QPROG=${SRC_DIR}/qexec.ott

usage () {
    echo " "
    echo "Usage: $0 [-p <program for query stream>] [-u1 <program for UF1>]"
    echo "          [-u2 <program for UF2>] [-o] [-s] [-h] [-u <user/password>]"
    echo " "
    echo "scale factor      : The scale factor of the run."
    echo "update ||ism      : The parallelism to use for the UFs."
    echo " "
    echo "-p <program>      : Program for Query Stream."
    echo "                  : Default is $QPROG."
    echo "-u1 <program>      : Program for UF1."
    echo "                  : Default is $U1PROG."
    echo "-u2 <program>      : Program for UF2."
    echo "                  : Default is $U2PROG."
    echo "-o                : Collect Oracle statistics."
    echo "-s                : Collect System statistics."
    echo "-u <user/passwd>   : User/Password. Default is tpcd/tpcd."
    echo "-h                : Displays this message."
    echo " "
    set -- `getopt "p:u1:u2:osu:h" "$@"` || usage
}

while :
do
    case "$1" in
        -u1) shift; U1PROG=$1;;
        -u2) shift; U2PROG=$1;;
        -p) shift; QPROG=$1;;
        -o) OSTAT=1;;
        -s) SSTAT=1;;
        -h) usage; exit 0;;
        --) shift; break;;
    esac
    shift;
done

if [ "$#" -ne "2" ]
then
    usage
    exit 1
fi

SF=$1
PARAM=$2

if [ $PARAM = first ];then
    UF_SET=1
else
    UF_SET=2
fi

if [ ! -f $RUN_ID_FILE ]
then
    echo "0" > $RUN_ID_FILE
fi

RUN_ID=`cat $RUN_ID_FILE`
RUN_ID=`expr $RUN_ID + 1`
export RUN_ID
echo $RUN_ID > $RUN_ID_FILE

echo "TPC-D Power Test Run `date`"
echo "RUNID is $RUN_ID"
echo " "

echo "TPC-D Power Test Run `date`" >> ${TPCD_RPT}/tpcd.${RUN_ID}.${HID}.rpt
echo "RUNID is $RUN_ID" >> ${TPCD_RPT}/tpcd.${RUN_ID}.${HID}.rpt
echo " " >> ${TPCD_RPT}/tpcd.${RUN_ID}.${HID}.rpt

TPCD_LOG_FILE=${TPCD_LOG}/query.${RUN_ID}.log
TPCD_RPT_FILE=${TPCD_RPT}/query.${RUN_ID}.rpt
QRY_FILE=${QRY_DIR}/qtemp.${RUN_ID}.${SF}

echo "Generates query template file for stream0"
echo "Generates query template file for stream0" >>
${TPCD_RPT}/tpcd.${RUN_ID}.${HID}.rpt
${QGEN} -c -p 0 -s ${SF} > ${QRY_FILE}
echo "Done generating query template file for stream0"
echo "Done generating query template file for stream0" >>
${TPCD_RPT}/tpcd.${RUN_ID}.${HID}.rpt

START=`GTIME`
echo "TPC-D Power Test Execution starts at $START"
echo "TPC-D Power Test Execution starts at $START" >>
${TPCD_RPT}/tpcd.${RUN_ID}.${HID}.rpt
echo " "
echo " " >> ${TPCD_RPT}/tpcd.${RUN_ID}.${HID}.rpt

# Execute UF1

SDATE=`date`
echo "Start UF1 at ${SDATE}"
echo "Start UF1 at ${SDATE}" >> ${TPCD_RPT}/tpcd.${RUN_ID}.${HID}.rpt
```

```

UF1_START=`${GTIME}`
echo "Start Time for UF1 is $UF1_START"
echo "Start Time for UF1 is $UF1_START" >> ${TPCD_RPT}/tpcd.${RUN_ID}.${HID}.rpt

${UPD_DIR}/audit_uf1.sh $UF_SET > ${OUT}/uf1.${RUN_ID}.${HID} 2>&1

# Execute Query Stream

UF1_END=`${GTIME}`
echo "End Time for UF1 is $UF1_END"
echo "End Time for UF1 is $UF1_END" >> ${TPCD_RPT}/tpcd.${RUN_ID}.${HID}.rpt

EDATE=`date`
echo "End UF1. Start Query Stream at ${EDATE}"
echo "End UF1. Start Query Stream at ${EDATE}" >>
${TPCD_RPT}/tpcd.${RUN_ID}.${HID}.rpt
echo "Start Query\t${EDATE}" >> ${TPCD_RPT}/tpcd.${RUN_ID}.${HID}.rpt

${QPROG} tpcd/tpcd q${QRY_FILE} l${TPCD_LOG_FILE} r${TPCD_RPT_FILE} \
> ${OUT}/qs.${RUN_ID}.${HID} 2>&1

# Execute UF2

SDATE=`date`
echo "End Query Stream. Start UF2 at ${SDATE}"
echo "End Query Stream. Start UF2 at ${SDATE}" >>
${TPCD_RPT}/tpcd.${RUN_ID}.${HID}.rpt
echo "Start UF2 at ${SDATE}" >> ${TPCD_RPT}/tpcd.${RUN_ID}.${HID}.rpt
UF2_START=`${GTIME}`
echo "Start Time for UF2 is $UF2_START"
echo "Start Time for UF2 is $UF2_START" >> ${TPCD_RPT}/tpcd.${RUN_ID}.${HID}.rpt

${UPD_DIR}/audit_uf2.sh $UF_SET > ${OUT}/uf2.${RUN_ID}.${HID} 2>&1

UF2_END=`${GTIME}`
echo "End Time for UF2 is $UF2_END"
echo "End Time for UF2 is $UF2_END" >> ${TPCD_RPT}/tpcd.${RUN_ID}.${HID}.rpt
EDATE=`date`
echo "END UF2 at ${EDATE}"
echo "END UF2 at ${EDATE}" >> ${TPCD_RPT}/tpcd.${RUN_ID}.${HID}.rpt

END=`$GTIME`

echo "TPC-D Power Test Execution ends at $END"
echo "TPC-D Power Test Execution ends at $END" >>
${TPCD_RPT}/tpcd.${RUN_ID}.${HID}.rpt
MEA_INT="echo $END - $START | bc"
echo "Measurement Interval is $MEA_INT"
echo " " >> ${TPCD_RPT}/tpcd.${RUN_ID}.${HID}.rpt
echo "Power Measurement Interval is $MEA_INT" >>
${TPCD_RPT}/tpcd.${RUN_ID}.${HID}.rpt

UF1_TIME="echo $UF1_END - $UF1_START | bc"
echo "Elapsed Time for Update Function 1 is $UF1_TIME"
echo "Elapsed Time for Update Function 1 is $UF1_TIME" >>
${TPCD_RPT}/tpcd.${RUN_ID}.${HID}.rpt

echo "-- UF1" >> ${TPCD_RPT_FILE}
echo "$UF1_TIME" >> ${TPCD_RPT_FILE}

UF2_TIME="echo $UF2_END - $UF2_START | bc"
echo "Elapsed Time for Update Function 2 is $UF2_TIME"
echo "Elapsed Time for Update Function 2 is $UF2_TIME" >>
${TPCD_RPT}/tpcd.${RUN_ID}.${HID}.rpt

echo "-- UF2" >> ${TPCD_RPT_FILE}
echo "$UF2_TIME" >> ${TPCD_RPT_FILE}

echo ""
echo "-----"
echo ""
echo "Power Test completed at `date`" >> ${TPCD_RPT}/tpcd.${RUN_ID}.${HID}.rpt
${SRC_DIR}/metric ${SF} < ${TPCD_RPT_FILE} \
>> ${TPCD_RPT}/tpcd.${RUN_ID}.${HID}.rpt

THRUPUT=`expr 17 \* 3600 \* ${SF}`
THRUPUT="echo `scale=2\n${THRUPUT}/${MEA_INT}` | bc"
echo "Real Throughput Metric is: $THRUPUT"
echo "Real Throughput Metric is: $THRUPUT" >> \
${TPCD_RPT}/tpcd.${RUN_ID}.${HID}.rpt

echo ""
echo "-----"
echo ""

audit_uf1.sh
#!/bin/ksh
#=====
#      Copyright (c) 1995 Oracle Corp, Redwood Shores, CA      |
#      OPEN SYSTEM PERFORMANCE GROUP                            |
#      All Rights Reserved                                       |
#=====
# FILENAME
#   audit_uf1.sh
# DESCRIPTION
#   audit_uf1.sh
# USAGE
#   To execute UF1.
#=====

UPDATE_DIR=/fs/fs_upd
SCRIPT_DIR=${UPDATE_DIR}/scripts
LOG_DIR=${UPDATE_DIR}/log
GTIME=${SCRIPT_DIR}/gtime
PASSWD=tpcd/tpcd
LDR_PAR=40

#
# Set SETNUM
#
if [ "$1" != "" ]; then

```

```

SETNUM=$1
else
if [ "$SETNUM" = "" ]; then
SETNUM=1
fi
fi

#
# Set PDDL/PDML degree
#
if [ "$PDML_DEGREE" = "" ]; then
PDML_DEGREE=160
fi

if [ "$PDDL_DEGREE" = "" ]; then
PDDL_DEGREE=64
fi

# first create the temp tables
load=1
if [ $load -eq 1 ]; then
svrmgr1 <<!
connect $PASSWD

alter session force parallel ddl parallel (degree $PDDL_DEGREE);
alter session force parallel dml parallel (degree $PDML_DEGREE);
alter session set isolation_level=serializable;

drop table temp_item;
create table temp_item (
l_shipdate          date ,
l_orderkey          number,
l_discount          number ,
l_extendedprice     number ,
l_suppkey           number,
l_quantity          number ,
l_returnflag        char(1) ,
l_partkey           number,
l_linestatus         char(1) ,
l_tax              number,
l_commitdate        date ,
l_receiptdate       date ,
l_shipmode          varchar2(10) ,
l_linenum           number,
l_shipinstruct       varchar2(25) ,
l_comment           varchar2(44)
)
parallel (degree $PDML_DEGREE instances 1)
nologging;

drop table temp_ord;
create table temp_ord (
o_orderdate         date ,
o_orderkey          number,
o_custkey           number,
o_orderpriority     varchar2(15) ,
o_shippriority      number ,
o_clerk             varchar2(15) ,
o_orderstatus       char(1) ,
o_totalprice        number ,
o_comment           varchar2(79)
)
parallel (degree $PDML_DEGREE instances 1)
nologging;

exit;

!

i=1
PID=""
START=`$GTIME`

#
# load into staging area
#
echo "load start"
date
while [ $i -le $LDR_PAR ]
do

sqlldr userid=$PASSWD control=${SCRIPT_DIR}/tempitem.ctl \
log=${LOG_DIR}/ti${i}.log \
data=${UPDATE_DIR}/u${SETNUM}/${LDR_PAR}/lineitem.tbl.u${SETNUM}.${i} \
direct=true parallel=true &

sqlldr userid=$PASSWD control=${SCRIPT_DIR}/tempord.ctl \
log=${LOG_DIR}/to${i}.log \
data=${UPDATE_DIR}/u${SETNUM}/${LDR_PAR}/order.tbl.u${SETNUM}.${i} \
direct=true parallel=true &

i=`expr $i + 1`

done

wait

fi

END_LOAD=`$GTIME`
echo "Load done. Elapsed Time is `echo $END_LOAD - $START | bc`"

# export LOAD_ONLY=yes

date

if [ "$LOAD_ONLY" = "yes" ]; then
exit
fi

# now do the insert

```

```

svrmgrl <<!

connect $PASSWD

set timing on;
set echo on

alter session force parallel ddl parallel (degree $PDDL_DEGREE);
alter session force parallel dml parallel (degree $PDML_DEGREE);
alter session set isolation_level=serializable;

analyze table temp_ord estimate statistics sample 100 rows;
analyze table temp_item estimate statistics sample 100 rows;

commit;

insert /*+ parallel(orders, $PDML_DEGREE, 1) */ into orders
(select * from temp_ord);
insert /*+ parallel(lineitem, $PDML_DEGREE, 1) */ into lineitem
(select * from temp_item);

commit;

exit;
!

wait

END=`$GTIME`

# Done

echo ""
echo "Update Function 1 Set $SETNUM done!"
echo "Elapsed Time is `echo $END - $START | bc`"
echo ""

audit_uf2.sh
#!/bin/ksh
#
#=====+
#      Copyright (c) 1995 Oracle Corp, Redwood Shores, CA      |
#      OPEN SYSTEM PERFORMANCE GROUP                          |
#      All Rights Reserved                                     |
#=====+
# FILENAME
#   uf2.sh
# DESCRIPTION
#   uf2.sh [-u <uid/passwd to login>] [-p <program>] <run_id>
#           <scale factor> <pair number> <parallelism>
# USAGE
#   To execute UF2.
#=====+
UPDATE_DIR=/fs/fs_upd
SCRIPT_DIR=${UPDATE_DIR}/scripts
LOG_DIR=${UPDATE_DIR}/log
GTIME=${SCRIPT_DIR}/gtime
PASSWD=tpcd/tpcd
LDR_PAR=40

#
# Set SETNUM
#
if [ "$1" != "" ]; then
  SETNUM=$1
else
  if [ "$SETNUM" = "" ]; then
    SETNUM=1
  fi
fi

#
# Set PDDL/PDML degree
#
if [ "$PDML_DEGREE" = "" ]; then
  PDML_DEGREE=160
fi

if [ "$PDDL_DEGREE" = "" ]; then
  PDDL_DEGREE=64
fi

# first create the temp tables
load=1
if [ $load -eq 1 ]; then
  svrmgrl <<!
    connect $PASSWD

    alter session force parallel ddl parallel (degree $PDDL_DEGREE);
    alter session force parallel dml parallel (degree $PDML_DEGREE);
    alter session set isolation_level=serializable;

    drop table temp_okey;
    create table temp_okey (
      t_orderkey      number)
    parallel (degree $PDML_DEGREE instances 1)
    nologging;

    exit;
  !

  i=1
  PID=""
  START=`$GTIME`

  #
  # load into staging area
  #

```

```

while [ $i -le $LDR_PAR ]
do

  sqlldr userid=$PASSWD control=${SCRIPT_DIR}/tempkey.ctl \
  log=${LOG_DIR}/ti${i}.log \
  data=${UPDATE_DIR}/u${SETNUM}/${LDR_PAR}/delete.${SETNUM}.${i} \
  direct=true parallel=true &

  i=`expr $i + 1`
done

wait

fi

END_LOAD=`$GTIME`
echo "Load done. Elapsed Time is `echo $END_LOAD - $START | bc`"

# export LOAD_ONLY=yes

if [ "$LOAD_ONLY" = "yes" ]; then
  exit
fi

svrmgrl <<!

connect $PASSWD ;

set timing on;
set echo on

alter session force parallel ddl parallel (degree $PDDL_DEGREE);
alter session force parallel dml parallel (degree $PDML_DEGREE);
alter session set isolation_level=serializable;

commit;

delete /*+ ordered use_nl(orders) index(orders, o_okey) */ orders
where o_orderkey in (select /*+ nl_sj */ * from temp_okey);

delete /*+ ordered use_nl(lineitem) index(lineitem, l_ored) */ lineitem
where l_orderkey in (select /*+ nl_sj */ * from temp_okey);

commit;

exit;
!

END=`$GTIME`

# Done

echo ""
echo "Update Function 2 Set $SETNUM done!"
echo "Elapsed Time is `echo $END - $START | bc`"
echo ""

gettime.c
#ifdef RCSID
static char *RCSid =
"$Header: gettime.c 7030100.1 96/05/21 15:31:36 plai Generic<base> $ Copyr (c)
1993 Oracle";
#endif /* RCSID */

/*=====+
|      Copyright (c) 1996 Oracle Corp, Redwood Shores, CA      |
|      OPEN SYSTEMS PERFORMANCE GROUP                          |
|      All Rights Reserved                                     |
|=====+
|
| FILENAME
| gettime.c
|
| ROUTINES
| gettime
| getcpu
| DESCRIPTION
| get wall clock time.
| get cpu time.
| NOTES
| Both routines return time in seconds as a double.
|=====+*/

/*
** Options:
**   TIME_W_TIMES:      implement gettime() with times().
**   TIME_W_GETTIME:    implement gettime() with gettimeofday().
**   CPU_W_TIMES:       implement getcpu() with times().
**   CPU_W_GETTRU:      implement getcpu() with getrusage().
**   GETRU_STATS:       collect getrusage statistics
**   GET_P_STATS:       collect get_process_stats statistics
*/

#define SUN_OS5

#ifdef SUN_OS5
#define TIME_W_GETTIME
#define CPU_W_TIMES
#define GETRU_STATS
#define CPU_W_GETRU
#endif /* SUN_OS5 */

#ifdef sequent || defined(SEQ_PXS)
#define GET_P_STATS
#endif /* sequent */

#ifdef aix || defined(AIXRIOS)
#define TIME_W_GETTIME
#define CPU_W_TIMES
#define GETRU_STATS
#endif /* AIXRIOS */

```

```

#if defined(a_osf) || defined(A_OSF)
# define TIME_W_GETTIME
# define CPU_W_GETRU
# define GETRU_STATS
#endif /* AIXRIOS */

#if defined(HPUX) || defined(XENIX_386) || defined(SYSV_386) || defined(ATT_3B)
# define TIME_W_TIMES
# define CPU_W_TIMES
#endif /* HPUX || XENIX_386 || SYSV_386 */

#if !defined(TIME_W_GETTIME) && !defined(TIME_W_TIMES)
# define TIME_W_TIMES
#endif

#if !defined(CPU_W_GETRU) && !defined(CPU_W_TIMES)
# define CPU_W_TIMES
#endif

#ifdef GET_P_STATS
# ifdef GETRU_STATS
# undef GETRU_STATS
# endif
#endif

#if defined(TIME_W_GETTIME) || defined(CPU_W_GETRU) || defined(GETRU_STATS)
# include <sys/time.h>
#endif /* TIME_W_GETTIME || CPU_W_GETRU || GETRU_STATS */

#if defined(CPU_W_GETRU) || defined(GETRU_STATS)
# include <sys/resource.h>
#endif /* CPU_W_GETRU || GETRU_STATS */

#if defined(TIME_W_TIMES) || defined(CPU_W_TIMES)
# include <sys/types.h>
# include <sys/times.h>
# include <sys/param.h> /* most systems define HZ here */
#endif /* TIME_W_TIMES or CPU_W_TIMES */

#ifdef GET_P_STATS
# include <sys/types.h>
# include <sys/procstats.h>
#endif /* GET_P_STATS */

# include <stdio.h>

#ifdef GETRU_STATS
struct rusage selfru;
struct rusage kidsru;
#endif /* GETRU_STATS */

#ifdef GET_P_STATS
struct process_stats selfru;
struct process_stats kidsru;
#endif /* GET_P_STATS */

double gettime ()
{
#ifdef TIME_W_GETTIME
    struct timeval tv;

    (void) gettimeofday (&tv, (struct timezone *) 0);
    return ((double) tv.tv_sec + (1.0e-6 * (double) tv.tv_usec));
#endif /* TIME_W_GETTIME */

#ifdef TIME_W_TIMES
    struct tms buf;

    return ((double) times (&buf) / HZ);
#endif /* TIME_W_TIMES */
}

double getcpu ()
{
#ifdef CPU_W_TIMES
    struct tms buf;

    (void) times (&buf);
    return (((double) buf.tms_utime + (double) buf.tms_stime) / HZ);
#endif /* CPU_W_TIMES */

#ifdef CPU_W_GETRU
    struct rusage ru;
    double usecs;

    (void) getrusage (0, &ru);
    usecs = 1.0e-6 * (double) (ru.ru_utime.tv_usec + ru.ru_stime.tv_usec);
    return ((double) (ru.ru_utime.tv_sec + ru.ru_stime.tv_sec) + usecs);
#endif /* CPU_W_GETRU */
}

getru (fp, kids, config, runname, proc_no)

FILE *fp;
int kids;
char *config;
char *runname;
int proc_no;

```

```

{
#ifdef GETRU_STATS
    struct rusage ru;

    fprintf (fp, "%-10.10s %-10.10s %10d %10d ", config, runname, proc_no, kids);
    getrusage (kids ? RUSAGE_CHILDREN : RUSAGE_SELF, &ru);
    print_ru (fp, &ru);
    fprintf (fp, "\n");
#endif /* GETRU_STATS */

#ifdef GET_P_STATS
    timeval_t tv;
    struct process_stats ru;

    fprintf (fp, "%-10.10s %-10.10s %10d %10d ", config, runname, proc_no, kids);
    if (kids)
        get_process_stats (&tv, PS_SELF, (struct process_stats *) 0, &ru);
    else
        get_process_stats (&tv, PS_SELF, &ru, (struct process_stats *) 0);
    print_ru (fp, &ru);
    fprintf (fp, "\n");
#endif /* GET_P_STATS */
}

getru1 (kids)

int kids;

{
#ifdef GETRU_STATS
    if (kids) {
        memset (&kidsru, 0, sizeof (kidsru));
        getrusage (RUSAGE_CHILDREN, &kidsru);
    }
    else {
        memset (&selfru, 0, sizeof (selfru));
        getrusage (RUSAGE_SELF, &selfru);
    }
#endif /* GETRU_STATS */

#ifdef GET_P_STATS
    timeval_t tv;

    if (kids) {
        memset (&kidsru, 0, sizeof (kidsru));
        get_process_stats (&tv, PS_SELF, (struct process_stats *) 0, &kidsru);
    }
    else {
        memset (&selfru, 0, sizeof (selfru));
        get_process_stats (&tv, PS_SELF, &selfru, (struct process_stats *) 0);
    }
#endif /* GET_P_STATS */
}

getru2 (fp, kids, config, runname, proc_no)

FILE *fp;
int kids;
char *config;
char *runname;
int proc_no;

{
#ifdef GETRU_STATS
    struct rusage ru;

    fprintf (fp, "%-10.10s %-10.10s %10d %10d ", config, runname, proc_no, kids);
    getrusage (kids ? RUSAGE_CHILDREN : RUSAGE_SELF, &ru);
    if (kids)
        diffrr (&ru, &kidsru);
    else
        diffrr (&ru, &selfru);
    print_ru (fp, &ru);
    fprintf (fp, "\n");
#endif /* GETRU_STATS */

#ifdef GET_P_STATS
    timeval_t tv;
    struct process_stats ru;

    fprintf (fp, "%-10.10s %-10.10s %10d %10d ", config, runname, proc_no, kids);
    if (kids)
        get_process_stats (&tv, PS_SELF, (struct process_stats *) 0, &ru);
    else
        get_process_stats (&tv, PS_SELF, &ru, (struct process_stats *) 0);
    if (kids)
        diffrr (&ru, &kidsru);
    else
        diffrr (&ru, &selfru);
    print_ru (fp, &ru);
    fprintf (fp, "\n");
#endif /* GET_P_STATS */
}

#ifdef GETRU_STATS
    print_ru (fp, ru)

```

```

FILE *fp;
struct rusage *ru;

{
    fprintf (fp, "%10ld ", ru->ru_etime.tv_sec * 1000 +
              (ru->ru_etime.tv_usec/1000));
    fprintf (fp, "%10ld ", ru->ru_stime.tv_sec * 1000 +
              (ru->ru_stime.tv_usec/1000));
    fprintf (fp, "%10ld ", ru->ru_maxrss);
    fprintf (fp, "%10ld ", ru->ru_majflt);
    fprintf (fp, "%10ld ", ru->ru_minflt);
    fprintf (fp, "%10ld ", 0);
    fprintf (fp, "%10ld ", 0);
    fprintf (fp, "%10ld ", 0);
    fprintf (fp, "%10ld ", ru->ru_nswap);
    fprintf (fp, "%10ld ", 0);
    fprintf (fp, "%10ld ", ru->ru_nvcsw);
    fprintf (fp, "%10ld ", ru->ru_nivcsw);
    fprintf (fp, "%10ld ", ru->ru_nsignals);
    fprintf (fp, "%10ld ", 0);
    fprintf (fp, "%10ld ", 0);
    fprintf (fp, "%10ld ", ru->ru_inblock);
    fprintf (fp, "%10ld ", ru->ru_oublock);
    fprintf (fp, "%10ld ", 0);
    fprintf (fp, "%10ld", 0);
}

diffrru (ru2, ru)

struct rusage *ru2;
struct rusage *ru;

{
    ru2->ru_etime.tv_sec -= ru->ru_etime.tv_sec;
    ru2->ru_etime.tv_usec -= ru->ru_etime.tv_usec;
    ru2->ru_stime.tv_sec -= ru->ru_stime.tv_sec;
    ru2->ru_stime.tv_usec -= ru->ru_stime.tv_usec;
    ru2->ru_maxrss -= ru->ru_maxrss;
    ru2->ru_ixrss -= ru->ru_ixrss;
    ru2->ru_idrss -= ru->ru_idrss;
    ru2->ru_minflt -= ru->ru_minflt;
    ru2->ru_majflt -= ru->ru_majflt;
    ru2->ru_nswap -= ru->ru_nswap;
    ru2->ru_inblock -= ru->ru_inblock;
    ru2->ru_oublock -= ru->ru_oublock;
    ru2->ru_msgsnd -= ru->ru_msgsnd;
    ru2->ru_msgrcv -= ru->ru_msgrcv;
    ru2->ru_nsignals -= ru->ru_nsignals;
    ru2->ru_nvcsw -= ru->ru_nvcsw;
    ru2->ru_nivcsw -= ru->ru_nivcsw;
}

#endif /* GETRU_STATS */

#ifdef GET_P_STATS

print_ru (fp, ps)

FILE *fp;
struct process_stats *ps;

{
    fprintf (fp, "%10lu ", ps->ps_etime.tv_sec * 1000 +
              (ps->ps_etime.tv_usec/1000));
    fprintf (fp, "%10lu ", ps->ps_stime.tv_sec * 1000 +
              (ps->ps_stime.tv_usec/1000));
    fprintf (fp, "%10lu ", ps->ps_maxrss);
    fprintf (fp, "%10lu ", ps->ps_pagein);
    fprintf (fp, "%10lu ", ps->ps_reclaim);
    fprintf (fp, "%10lu ", ps->ps_zerofill);
    fprintf (fp, "%10lu ", ps->ps_pffincr);
    fprintf (fp, "%10lu ", ps->ps_pffdecr);
    fprintf (fp, "%10lu ", ps->ps_swap);
    fprintf (fp, "%10lu ", ps->ps_syscall);
    fprintf (fp, "%10lu ", ps->ps_volcsw);
    fprintf (fp, "%10lu ", ps->ps_involcsw);
    fprintf (fp, "%10lu ", ps->ps_signal);
    fprintf (fp, "%10lu ", ps->ps_lread);
    fprintf (fp, "%10lu ", ps->ps_lwrite);
    fprintf (fp, "%10lu ", ps->ps_bread);
    fprintf (fp, "%10lu ", ps->ps_bwrite);
    fprintf (fp, "%10lu ", ps->ps_phread);
    fprintf (fp, "%10lu", ps->ps_phwrite);
}

diffrru (ru2, ru)

struct process_stats *ru2;
struct process_stats *ru;

{
    ru2->ps_etime.tv_sec -= ru->ps_etime.tv_sec;
    ru2->ps_etime.tv_usec -= ru->ps_etime.tv_usec;
    ru2->ps_stime.tv_sec -= ru->ps_stime.tv_sec;
    ru2->ps_stime.tv_usec -= ru->ps_stime.tv_usec;
    ru2->ps_maxrss -= ru->ps_maxrss;
    ru2->ps_pagein -= ru->ps_pagein;
    ru2->ps_reclaim -= ru->ps_reclaim;
    ru2->ps_zerofill -= ru->ps_zerofill;
    ru2->ps_pffincr -= ru->ps_pffincr;
    ru2->ps_pffdecr -= ru->ps_pffdecr;
    ru2->ps_swap -= ru->ps_swap;
    ru2->ps_syscall -= ru->ps_syscall;
    ru2->ps_volcsw -= ru->ps_volcsw;
    ru2->ps_involcsw -= ru->ps_involcsw;
    ru2->ps_signal -= ru->ps_signal;
    ru2->ps_lread -= ru->ps_lread;
    ru2->ps_lwrite -= ru->ps_lwrite;
    ru2->ps_bread -= ru->ps_bread;
    ru2->ps_bwrite -= ru->ps_bwrite;
    ru2->ps_phread -= ru->ps_phread;
    ru2->ps_phwrite -= ru->ps_phwrite;
}

#endif /* GET_P_STATS */

#endif /* GET_P_STATS */

gtime.c
/*
 * Copyright (c) 1996 Oracle Corp, Redwood Shores, CA
 * OPEN SYSTEMS PERFORMANCE GROUP
 * All Rights Reserved
 */
=====
| FILENAME
| gtime.c
| DESCRIPTION
| Wrapper for gettimeofday
|=====
/*

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

double gettimeofday();

main() {
    printf("%f", gettimeofday());
    exit(0);
}

gexecpl.c
/*
 * Copyright (c) 1997 Oracle Corp, Redwood Shores, CA
 * OPEN SYSTEMS PERFORMANCE GROUP
 * All Rights Reserved
 */
=====
| FILENAME
| gexecpl.c
| DESCRIPTION
| SQL Execution Engine, Oracle v8, OCI version
| MODIFIED
| pswong 02/19/97 - migrating to version 8
| pswong 04/02/96 - more polishing
| pswong 03/25/96 - polish up
| pswong03/06/96 - created
|=====
/*

#include <stdio.h>
#include <string.h>
#include <setjmp.h>
#include <sys/param.h>
#include <errno.h>
#include <math.h>
#include <string.h>
#include <sys/types.h>
#include <time.h>

#include "gexecpl.h"

/* Function Prototypes */

extern double gettimeofday();
extern char *getenv();

/* function prototypes from gen.c */

int get_statement();

/* Declare error handling functions */

void sql_error();

/* Other prototypes */

int define_output_variables();
void process_select_list();
void usage();
void SQLinit();
void SQLexec();
void SQLexit();
void *memalloc();
void print_header();
void print_rows();
int OPEN();
void remove_newline();

char logname[UNAME_LEN]; /* username/passwd combo */
char *passwd;

double tr_start = 0.0; /* query start time */
double tr_end = 0.0; /* query end time */

double s_tr_start = 0.0; /* statement start time */
double s_tr_end = 0.0; /* statement end time */

/* For our purpose of timing, we will treat comments as delimiters */
/* for queries. Thus, we will collect query timings whenever we */
/* encounter a comment (of course not for the first comment in a */

```

```

/* file). */

int end_flag = 0; /* flag to indicate that we have reached */
/* the end of a query */

int stmt_cnt = 0; /* Number of statements processed. */
int qry_cnt = 0; /* Number of query processed. */

double product = 1.0; /* cumulative product of query times */
int rows_ret = 0; /* the number of rows fetched */
int num_sel_list = 0; /* the number of select list item */

long num_to_fetch = -1; /* Number of rows to fetch. -1 means fetch all */

sltype slist[MAX_SEL_LIST]; /* Array for describing Select List */
dltype *dlist[MAX_SEL_LIST]; /* Array of ptrs for Defining Select List */

char stmt[SQL_LEN]; /* The SQL statement or comment line. */
char cmnt[81]; /* Buffer to save the comment. */

FILE *qtemp = stdin; /* fd for query template */
FILE *logfile = stdout; /* log and report files */
FILE *rep = stdout;

void *defbuf; /* Buffer pointer for ODEFIN */
int deflen = 0; /* Size of data type for ODEFIN */
int deftype = 1; /* Oracle type number for ODEFIN */

int pfmem = PFMEMSIZE; /* Memory to prefetch rows */

time_t tim; /* To get wall clock time */

/* OCI handles */

OCIServer *tpcsrv = NULL;
OCISvcCtx *tpcsvc = NULL;
OCISession *tpcusr = NULL;
OCISstmt *curq = NULL;
OCISstmt *curi = NULL;
OCIParm *tpcpar = NULL;

sword status = OCI_SUCCESS; /* OCI return value */

/* usage: prints the usage of the program */

void usage() {
    fprintf(stderr, "\nUsage: qexec.ott username/password [q-path name for query\n");
    fprintf(stderr, "template file]\n");
    fprintf(stderr, " [l-path name for log] [r-path name for\n");
    fprintf(stderr, "reports]\n");
    fprintf(stderr, "Options:\n");
    fprintf(stderr, "q-path for query : full path name for the query\n");
    fprintf(stderr, "template file.\n");
    fprintf(stderr, "l-path name for log : (default is stdin)\n");
    fprintf(stderr, "r-path name for reports : (default is stdout)\n");
    fprintf(stderr, "full path name for reports : (default is stdout)\n");
    exit(-1);
}

/* type: 0 if environment handle is passed, 1 if error handle is passwd */

void sql_error(errhp, status, type)
OCIServer *errhp;
sword status;
sword type;
{
    char msg[2048];
    ub4 errcode;
    ub4 msglen;
    int i, j;

    switch(status) {
        case OCI_SUCCESS_WITH_INFO:
            fprintf(stderr, "Error: Statement returned with info.\n");
            if (type)
                (void) OCIServerGet(errhp, 1, NULL, (sb4*)&errcode, (text*)msg,
                2048, OCI_HTYPE_ERROR);
            else
                (void) OCIServerGet(errhp, 1, NULL, (sb4*)&errcode, (text*)msg,
                2048, OCI_HTYPE_ENV);
            fprintf(stderr, "%s\n", msg);
            break;
        case OCI_ERROR:
            fprintf(stderr, "Error: OCI call error.\n");
            if (type)
                (void) OCIServerGet(errhp, 1, NULL, (sb4*)&errcode, (text*)msg,
                2048, OCI_HTYPE_ERROR);
            else
                (void) OCIServerGet(errhp, 1, NULL, (sb4*)&errcode, (text*)msg,
                2048, OCI_HTYPE_ENV);
            fprintf(stderr, "%s\n", msg);
            break;
        case OCI_INVALID_HANDLE:
            fprintf(stderr, "Error: Invalid Handle.\n");
            if (type)
                (void) OCIServerGet(errhp, 1, NULL, (sb4*)&errcode, (text*)msg,
                2048, OCI_HTYPE_ERROR);
            else
                (void) OCIServerGet(errhp, 1, NULL, (sb4*)&errcode, (text*)msg,
                2048, OCI_HTYPE_ENV);
            fprintf(stderr, "%s\n", msg);
            break;
    }
}

/* Rollback just in case */

(void) OCITransRollback(tpcsvc, errhp, OCI_DEFAULT);

fprintf(stderr, "Exiting Oracle...\n");
fflush(stderr);

SQLexit();

exit(1);
}

void main(argc, argv)
int argc;
char *argv[];
{
    int i;
    int retcode; /* Return code for get_statement */

    /* Initialize some variables */

    if ((argc > 5) || (argc < 2)) {
        usage();
    }

    /* argv[1] -- User and Password for Database */

    strcpy(logname, argv[1]);

    /* Process optional parameters */

    argc -= 1;
    argv += 1;

    while(--argc) {
        ++argv;
        switch(argv[0][0]) {
            case 'q':
                if ((qtemp = fopen(++argv[0], "r")) == NULL) {
                    fprintf(stderr, "Unable to open file '%s'\n", argv[0]);
                    fprintf(stderr, "%s: %s\n", argv[0], strerror(errno));
                    exit(-1);
                }
                break;
            case 'r':
                if ((rep = fopen(++argv[0], "a")) == NULL) {
                    fprintf(stderr, "Unable to open file '%s'\n", argv[0]);
                    fprintf(stderr, "%s: %s\n", argv[0], strerror(errno));
                    exit(-1);
                }
                break;
            case 'l':
                if ((logfile = fopen(++argv[0], "a")) == NULL) {
                    fprintf(stderr, "Unable to open file '%s'\n", argv[0]);
                    fprintf(stderr, "%s: %s\n", argv[0], strerror(errno));
                    exit(-1);
                }
                break;
            default:
                fprintf(stderr, "Invalid Option: %c\n", argv[0][0]);
                usage();
                break;
        }
    }

    /* Do some initialization and establish connection with the database */

    SQLinit();

    /* May want to add some triggering mechanism here */

    time(&tim);
    fprintf(logfile, "Begin Execution at %s\n", ctime(&tim));

    /* Get the next statement and start processing it */

    while ((retcode = get_statement()) > 0) {
        switch (retcode) {
            /* If this is a comment, skips it */
            case COMMENT:
                if (end_flag) {
                    /* reset query end flag */
                    end_flag = 0;
                    /* save the comment so that we can print it out later on */
                    strcpy(cmnt, stmt);
                    break;
                }
                fprintf(logfile, "%s", stmt);
                fprintf(rep, "%s", stmt);
                break;
            /* if this is a set_row_fetch command */
            case SET_FETCHROW:
                fprintf(logfile, "Setting the number of rows to fetch to: %ld\n",
                num_to_fetch);
                break;
            /* if this is a SQL statement */
            case SQL_STMT:
                /* Executes the query */

                SQLexec();

                s_tr_end = gettimeofday();
                stmt_cnt++;

                /*
                fprintf(logfile, "\nStatement Started at %.2f\n", s_tr_start);

```

```

fprintf(logfile,"Statement Ended at %.2f\n", s_tr_end);
*/
fprintf(logfile,"Statement Processed in %.2f seconds.\n",
(s_tr_end - s_tr_start));
break;

/* Should never reach here */
default:
fprintf(stderr, "Invalid statement type!!\n");
SQLexit();
break;
}
}

/* Get Timing for the last query */
tr_end = gettimeofday();

time(&tim);
fprintf(logfile,"nEnded Executing this Query at %s\n", ctime(&tim));
fprintf(logfile,"nQuery Started at %.2f\n", tr_start);
fprintf(logfile,"Query Ended at %.2f\n", tr_end);
fprintf(logfile,"Query Processed in %.2f seconds\n\n",
(tr_end - tr_start));

fprintf(rep, "%.2f\n", (tr_end - tr_start));

fprintf(logfile, "nSQL statements processed: %d\n", stmt_cnt);
fprintf(logfile, "Queries processed: %d\n", qry_cnt);

fflush(rep);
fflush(logfile);

/* Close the query template file */
fclose(qtemp);

/* Disconnect from ORACLE. */
SQLexit();
exit(0);
}

/* SQLinit(): Perform initialization tasks. */
/* Logs on to Oracle, opens some files and open a cursor for */
/* later use. */

void SQLinit()
{
int i;
char *pddl_degree;
char *pdml_degree;

/* preallocate MAX_PREALLOC members of the dlist array */
/* initializes others to NULL so that we can determine who to free later */

for (i=0; i<MAX_SEL_LIST; i++) {
if (i < MAX_PREALLOC) {
dlist[i] = (dlist *) malloc (sizeof(dlist));
dlist[i]->defhdl = NULL;
/* OCIthalloc(curq,&(dlist[i]->defhdl),OCI_HTYPE_DEFINE); */
}
else
dlist[i] = NULL;
}

/* Connect to ORACLE. Program will call sql_error() */
/* if an error occurs in connecting to the default database. */

(void) OCIInitialize(OCI_DEFAULT, (void *)0,0,0,0);

if ((status=OCIEnvInit((OCIEnv **) &tpcenv,OCI_DEFAULT,0, (void **)0)) !=
OCI_SUCCESS)
sql_error(tpcenv, status, 0);

OCIthalloc(tpcenv,&errhp,OCI_HTYPE_ERROR);
OCIthalloc(tpcenv,&curq,OCI_HTYPE_STMT);
OCIthalloc(tpcenv,&curi,OCI_HTYPE_STMT);
OCIthalloc(tpcenv,&tpcsvc,OCI_HTYPE_SVCCTX);
OCIthalloc(tpcenv,&tpcsrv,OCI_HTYPE_SERVER);
OCIthalloc(tpcenv,&tpcsur,OCI_HTYPE_SESSION);

/* get username and password */

passwd = strchr(logname, '/');
*passwd = '\0';
passwd++;

if ((status = OCIServerAttach(tpcsrv,errhp,(text *)0,0,OCI_DEFAULT)) !=
OCI_SUCCESS)
sql_error(errhp,status,1);

OCIaset(tpcsvc,OCI_HTYPE_SVCCTX,tpcsrv,0,OCI_ATTR_SERVER,errhp);
OCIaset(tpcsur,OCI_HTYPE_SESSION,logname,strlen(logname),OCI_ATTR_USERNAME,
errhp);
OCIaset(tpcsur,OCI_HTYPE_SESSION,passwd,strlen(passwd),OCI_ATTR_PASSWORD,
errhp);

if ((status = OCISessionBegin(tpcsvc, errhp, tpcsur, OCI_CRED_RDBMS,
OCI_DEFAULT)) != OCI_SUCCESS)
sql_error(errhp,status,1);

OCIaset(tpcsvc,OCI_HTYPE_SVCCTX,tpcsur,0,OCI_ATTR_SESSION,errhp);

/* Enable session parallel dml */

if (getenv((const char *) "PDML_DEGREE") != (char *)0)
pdml_degree = getenv((const char *) "PDML_DEGREE");
else

pdml_degree = "80";

sprintf((char *) stmt, PDMLTXT, pdml_degree);
OCISetPrepare(cur_i,errhp,(text *)stmt,strlen((char *)stmt),
OCI_NT_V_SYNTAX,OCI_DEFAULT);
OCISexec(tpcsvc,cur_i,errhp,1);

if (getenv((const char *) "PDDL_DEGREE") != (char *)0)
pddl_degree = getenv((const char *) "PDDL_DEGREE");
else
pddl_degree = "8";

sprintf((char *) stmt, PDDLTXT, pddl_degree);

OCISetPrepare(cur_i,errhp,(text *)stmt,strlen((char *)stmt),
OCI_NT_V_SYNTAX,OCI_DEFAULT);
OCISexec(tpcsvc,cur_i,errhp,1);

/*
if ((status=OCILogon((OCIEnv *)tpcenv,(OCIError *)errhp,(OCISvcCtx *)tpcsvc,
(text *)logname, strlen(logname), (text *)passwd,
strlen(passwd), (text *)0, 0)) != OCI_SUCCESS)
sql_error(errhp, status, 1);
*/
printf("nConnected to ORACLE as user: %s\n", logname);
}

/* SQLexec() Executes the SQL statement. */
/* Parse the SQL statement. */
/* If DDL or DML statements, execute right away. */
/* Else describe and define select list outputs, */
/* execute and fetch results. */

void SQLexec()
{
int i;
ub2 stmttyp = OCI_STMT_SELECT; /* default is a SELECT statement */

if (!end_flag) {

/* Clause 5.3.6.2: QI(i,s) is the time between the first character */
/* of this query text is submitted and the first */
/* character of the next query text is submitted. */

tr_end = gettimeofday();

if (qry_cnt) {
time(&tim);
fprintf(logfile,"nEnded Executing this Query at %s\n", ctime(&tim));
fprintf(logfile,"nQuery Started at %.2f\n", tr_start);
fprintf(logfile,"Query Ended at %.2f\n", tr_end);
fprintf(logfile,"Query Processed in %.2f seconds.\n\n",
(tr_end - tr_start));

fprintf(logfile,"-----\n\n");

/* print comments for this query that we have saved */

fprintf(logfile, "%s\n", cmnt);

fprintf(rep, "%.2f\n", (tr_end - tr_start));
fprintf(rep, "%s", cmnt);

fprintf(logfile,"nBegan Executing this Query at %s\n", ctime(&tim));

/* Let's fflush stuff so that we can see what's going on */

fflush(logfile);
fflush(rep);

}

tr_start = tr_end;
qry_cnt++;

end_flag = 1;

s_tr_start = gettimeofday();

/* prepare the statement */

if ((status = OCISetPrepare(curq, errhp, (text *) stmt, (ub4) strlen(stmt),
OCI_NT_V_SYNTAX, OCI_DEFAULT)) != OCI_SUCCESS)
sql_error(errhp,status,1);

/* Prints the query text to the logfile */

fprintf(logfile, "n%s\n", stmt);

/* if this is a DDL or DML statement, execute it right away */
/* only worries about SELECT statements right now, cannot */
/* execute a stored PL/SQL procedure in this version */

OCIaset(curq,OCI_HTYPE_STMT,&stmttyp,NULL,OCI_ATTR_STMT_TYPE,errhp);

if (stmttyp != OCI_STMT_SELECT) {
OCISexec(tpcsvc,curq,errhp,1);
return;
}

/* otherwise, this is a select statement */
/* Describe and define output variables */

/* first let's execute it to get the select-list definition */

OCIaset(curq, OCI_HTYPE_STMT, &pfmem, 0, OCI_ATTR_PREFETCH_MEMORY, errhp);

```

```

OCISeexec(tpcscv, curq, errhp, 0);

num_sel_list = define_output_variables();

/* Executes the query and fetches the rows */
(void) process_select_list(num_sel_list);

/* Need to get the number of rows fetched first
 * since the following statements will screw it up */
OCIaget(curq, OCI_HTYPE_STMT, &rows_ret, NULL, OCI_ATTR_ROW_COUNT, errhp);

/* To control memory usage, let's free up the extra dlist entries */
/* that we have allocated. */

i=MAX_PREALLOC;
while(dlist[i] != NULL) {
    free(dlist[i]);
    dlist[i++] = NULL;
}

/* reset set_fetchrows */
num_to_fetch = -1;
}

void SQLexit() {
    int i;

    OCILogoff(tpcscv, errhp);
    OCIhfree(tpcenv, OCI_HTYPE_STMT);
    OCIhfree(tpcscv, OCI_HTYPE_SVCCTX);
    OCIhfree(tpcscr, OCI_HTYPE_SERVER);
    OCIhfree(tpcusr, OCI_HTYPE_SESSION);

    /* free all memory */

    for (i=0; i<MAX_SEL_LIST; i++) {
        if (dlist[i] != NULL) {
            free(dlist[i]);
        }
    }

    /* Flush all output */

    fflush(rep);
    fflush(logfile);
}

/* define_output_variables(): Describe and define select-list items for */
/* a query statement. */
/* Returns the number of select-list items */
/* for this query. */

int define_output_variables()
{
    int i;
    int retflag = 0;

    for (i=0; i<MAX_SEL_LIST; i++) {
        slist[i].buflen = MAX_COLNAME_SIZE;

        if (OCIParamGet(curq, OCI_HTYPE_STMT, errhp, (dvoid **) &tpcpar,
            POS(i)) != OCI_SUCCESS)
            break;

        /* dsize and nullok fields of dlist not used */

        OCIaget(tpcpar, OCI_DTYPE_PARAM, &(slist[i].dsize),
            NULL, OCI_ATTR_DATA_SIZE, errhp);
        OCIaget(tpcpar, OCI_DTYPE_PARAM, &(slist[i].dbtype),
            NULL, OCI_ATTR_DATA_TYPE, errhp);
        OCIaget(tpcpar, OCI_DTYPE_PARAM, &(slist[i].buf),
            &(slist[i].buflen), OCI_ATTR_NAME, errhp);
        OCIaget(tpcpar, OCI_DTYPE_PARAM, &(slist[i].precision),
            NULL, OCI_ATTR_PRECISION, errhp);
        OCIaget(tpcpar, OCI_DTYPE_PARAM, &(slist[i].scale),
            NULL, OCI_ATTR_SCALE, errhp);

        /* For formatting purpose, remove trailing blanks in select-list name. */

        if (slist[i].buflen < MAX_COLNAME_SIZE)
            (slist[i].buf)[slist[i].buflen] = '\0';
    }

    /* Well, we need to allocate for entries for dlist */

    if (i >= MAX_PREALLOC) {
        dlist[i] = (dtype *) memalloc(sizeof(dtype));
        dlist[i]->defhdl = NULL;
    }

    /* Let's check the sizes and types for this select list item */

    switch (slist[i].dbtype) {
        case OCI_TYPECODE_NUMBER:

            /* The odescr will not give a good estimate to the scale if */
            /* no scale was given in the Oracle table definition. */

#ifdef HAVE_SCALE
            if (slist[i].scale != 0) {
                if (slist[i].scale != 0) {
                    defbuf = (double *) dlist[i]->fbuf;
                    deflen = FLT;
                    deftype = OCI_TYPECODE_DOUBLE;
                    slist[i].dbtype = OCI_TYPECODE_DOUBLE;
                } else {
                    defbuf = (int *) dlist[i]->ibuf;
                    deflen = INT;
                    deftype = OCI_TYPECODE_INTEGER;
                    slist[i].dbtype = OCI_TYPECODE_INTEGER;
                }
            }
            #else
            defbuf = (double *) dlist[i]->fbuf;
            deflen = FLT;
            deftype = OCI_TYPECODE_FLOAT;
            slist[i].dbtype = OCI_TYPECODE_FLOAT;
            #endif /* HAVE_SCALE */

            break;

            default:

                /* default is character string */

                defbuf = (char **) dlist[i]->sbuf;
                deflen = MAX_STR_LEN;
                deftype = SQLT_STR;
                /* deftype = OCI_TYPECODE_CHAR; */
                break;
            }

            /* Define the column */

            if ((status=OCIDefineByPos(curq, &(dlist[i]->defhdl), errhp, POS(i),
                defbuf, deflen, deftype, NULL,
                dlist[i]->rlen, NULL, OCI_DEFAULT)) != OCI_SUCCESS)
                sql_error(errhp, status, i);
        }
        return i;
    }

    /* process_select_list(): Fetch rows from a query. */

    void process_select_list(num)
        int num; /* number of select list items */
    {
        int i, j;
        int ntf;
        int num_so_far;
        sword stats = OCI_SUCCESS;

        /* Print the headers for the query execution result */

        print_header(num);

        /* See if we need to limit the rows to fetch */

        ntf = (num_to_fetch >= 0) ? num_to_fetch : MAX_ARRAY;

        /* Fetch the rows and print them out */

        if ((ntf > MAX_ARRAY) || (num_to_fetch == -1)) {
            stats = OCISTmtFetch(curq, errhp, MAX_ARRAY, OCI_FETCH_NEXT, OCI_DEFAULT);

            OCIaget(curq, OCI_HTYPE_STMT, &rows_ret, NULL, OCI_ATTR_ROW_COUNT, errhp);

            print_rows(num, rows_ret);

            /* To avoid 1022 from OPEN */
            /* More rows to fetch... */

            if (stats != OCI_NO_DATA) {
                if (num_to_fetch == -1) {
                    while ((stats = OCISTmtFetch(curq, errhp, MAX_ARRAY, OCI_FETCH_NEXT,
                        OCI_DEFAULT)) == OCI_SUCCESS) {
                        OCIaget(curq, OCI_HTYPE_STMT, &num_so_far, NULL,
                            OCI_ATTR_ROW_COUNT, errhp);
                        print_rows(num, (num_so_far-rows_ret));
                        rows_ret = num_so_far;
                    }
                }
                /* Print the final rows */
                OCIaget(curq, OCI_HTYPE_STMT, &num_so_far, NULL,
                    OCI_ATTR_ROW_COUNT, errhp);
                print_rows(num, (num_so_far-rows_ret));
                rows_ret = num_so_far;
            } else {
                ntf = MAX_ARRAY;
            }
        }

        while ((stats = OCISTmtFetch(curq, errhp,
            ((ntf > MAX_ARRAY) ? MAX_ARRAY : ntf),
            OCI_FETCH_NEXT, OCI_DEFAULT)) ==
            OCI_SUCCESS) {
            ntf = MAX_ARRAY;
            OCIaget(curq, OCI_HTYPE_STMT, &num_so_far, NULL,
                OCI_ATTR_ROW_COUNT, errhp);
            print_rows(num, (num_so_far-rows_ret));
            rows_ret = num_so_far;
            if (ntf <= 0) break;
        }
        OCIaget(curq, OCI_HTYPE_STMT, &num_so_far, NULL,
            OCI_ATTR_ROW_COUNT, errhp);
        print_rows(num, (num_so_far-rows_ret));
        rows_ret = num_so_far;
    }
}

```

```

OCIaget(curq,OCI_HTYPE_STMT,&rows_ret,NULL,OCI_ATTR_ROW_COUNT,errhp);
print_rows(num,rows_ret);
}

fprintf(logfile, "\n\n%d row%c processed.\n", rows_ret,
rows_ret == 1 ? '\0' : 's');
}

int get_statement()
{
char line[128];
char *pos, *str;

/* Reset statement buffer */

stmt[0] = '\0';

while (fgets(line, 127, qtmp) != NULL) {

/* skip blank lines */
if (line[0] == '\n')
continue;

/* remove blanks */

str = line;

while (*str == ' ') str++;

/* Let's get the line together first */

strcat(stmt, str);

/* if this is a comment line */
if ((str[0] == '-') && (str[1] == '-'))
return COMMENT;

/* see if this is a set_fetchrows line */
if (strncmp(str, "set_fetchrows", 13) == 0) {
pos = strchr(str, ';');
*pos = '\0';
pos = strchr(str, '=');
num_to_fetch = atol(++pos);
return SET_FETCHROW;
}

/* if this is the end of the current statement */
if ((pos = strchr(stmt, ';')) != NULL) {
*pos = '\0';
return SQL_STMT;
}
}
return END_OF_FILE;
}

/* memalloc(): Allocates memory, exit program if we have a problem. */

void *memalloc(size)
int size;
{
void *tmp;

if ((tmp = (void *) malloc(size)) == NULL) {
fprintf(stderr, "Error in malloc\n");
SQLexit();
return NULL; /* should never reach here */
} else {
return tmp;
}
}

void print_header(nsel)
int nsel; /* Number of select list items */
{
int i, diff;
char colname[MAX_COLNAME_SIZE];
int len = 0; /* Running column length */
int cwid = 0;

fprintf(logfile, "\n");

for (i=0; i<nsel; i++) {

/* extract the column name */

strncpy((char *)colname, (char *)slist[i].buf, slist[i].buflen);
colname[slist[i].buflen] = '\0';

/* format the output a little */

cwid = MAX(slist[i].dbsize, slist[i].buflen);

/* do a little bit of formatting */

if (cwid > 80) {
fprintf(logfile, "\n");
len = 0;
} else if ((len += cwid) > 80) {
fprintf(logfile, "\n");
len = cwid;
}
}
#ifdef FORMAT1
if ((slist[i].dbtype == INT_TYPE) || (slist[i].dbtype == FLT_TYPE))

```

```

fprintf(logfile, "%s ", cwid, slist[i].buf);
else /* string type */
fprintf(logfile, "%s ", -cwid, slist[i].buf);
#else
fprintf(logfile, "%s ", -cwid, colname);
#endif /* FORMAT1 */
}

fprintf(logfile, "\n");
}

```

```

void print_rows(ncol, nrow)
int ncol;
int nrow;
{
int i,j;
int len;
int diff;
int cwid;

for (i=0; i<nrow; i++) {
len = 0;

for (j=0; j<ncol; j++) {
cwid = MAX(slist[j].dbsize, slist[j].buflen);

/* do a little bit of formatting */

if (cwid > 80) {
fprintf(logfile, "\n");
len = 0;
} else if ((len += cwid) > 80) {
fprintf(logfile, "\n");
len = cwid;
}

switch(slist[j].dbtype) {
case INT_TYPE:
#ifdef HAVE_SCALE
fprintf(logfile, "%*ld|", cwid, (dlist[j]->ibuf)[i]);
break;
#endif /* HAVE_SCALE */
case FLT_TYPE:
#ifdef FORMAT1
fprintf(logfile, "%*.2f ", cwid, (dlist[j]->fbuf)[i]);
#else
fprintf(logfile, "%*.2f ", -cwid, (dlist[j]->fbuf)[i]);
#endif /* FORMAT1 */
break;
default:
fprintf(logfile, "%s ", -(cwid), (dlist[j]->sbuf)[i]);
break;
}
}
fprintf(logfile, "\n");
}
}

```

```

/* remove_newline(): Remove newline character from str. */

void remove_newline(str)
char *str;
{
char *p;

while ((p = strchr(str, '\n')) != NULL)
*p = ' ';
}

```

```

gexecpl.h
/*=====
|      Copyright (c) 1995 Oracle Corp, Redwood Shores, CA
|      OPEN SYSTEMS PERFORMANCE GROUP
|      All Rights Reserved
|=====
| FILENAME
|      qstream.h
| DESCRIPTION
|      SQL statement execution front-end header file.
| MODIFIED
|      pswong      03/07/96 - created
|=====*/

```

```

#ifdef QSTREAMPL_H

#define QSTREAMPL_H

#include <stdio.h>
#include <string.h>
#include <sys/param.h>
#include <sys/types.h>
#include <time.h>
#include <errno.h>
#include <math.h>

#include <oratypes.h>

#include <oratypes.h>

#ifdef OCIDFN
#include <ocidfn.h>
#endif /* OCIDFN */

#ifdef OCI_ORACLE

```

```

#include <oci.h>
#endif /* OCI_ORACLE */
/*
#ifdef __STDC__
#include <ociapr.h>
#else
#include <ocikpr.h>
#endif /* __STDC__ */

/* some basic definitions */

#define UNAME_LEN 64
#define MAX_FILE_PATH_LEN 128

#ifdef TRUE
#define TRUE 1
#endif /* TRUE */

#ifdef FALSE
#define FALSE 1
#endif /* FALSE */

#define MAX(x,y) ((x >= y) ? x : y)
#define MIN(x,y) ((x <= y) ? x : y)

/* defines and typedefs for parsing */

#define CRT_TBL 1
#define INS_STMT 3
#define SEL_STMT 4
#define UPD_STMT 5
#define DRP_VIEW 7
#define DRP_TBL 8
#define DEL_STMT 9
#define CRT_VIEW 10

/* defines and typedefs for query description */

#define MAX_COLNAME_SIZE 32 /* Maximum length of Column name */
#define MAX_SEL_LIST 16 /* Maximum items on a select list */

#define END_OF_LIST 1007 /* Error code when we reach the end of the */
/* select list. */

/* types for describe */

#define CHAR_TYPE 1
#define NUM_TYPE 2
#define INT_TYPE 3
#define FLT_TYPE 4
#define STR_TYPE 5
#define DATE_TYPE 12

#define NUMWIDTH 16 /* Width of the numeric fields */

#define POS(i) (i+1) /* The position is 1..n instead */
#define IND(i) (i-1) /* of 0..n-1 as in an array. */

typedef struct des
{
    ub2 dbsize;
    ub4 buflen;
    /* sb2 dsize; */
    sb4 scale;
    /* sb2 nullok; */
    OCITextCode dbtype;
    /* text buf[MAX_COLNAME_SIZE]; */
    text *buf;
    ub1 precision;
} sltype;

/* defines and typedefs for query select list definition */

#define MAX_ARRAY 1000 /* Maximum array size for array fetch */
#define PFMEMSIZE 65536 /* Memory size of prefetch buffer */

#define MAX_STR_LEN 256 /* Maximum size for string variables */
#define MAX_PREALLOC 8 /* Maximum number of preallocated select list */
/* definitions. */

#define INT sizeof(long)
#define STR sizeof(char)
#define FLT sizeof(double)

#define FLTP (double *)
#define INTP (long *)
#define STRP (char **)

typedef struct def
{
    long ibuf[MAX_ARRAY];
    double fbuf[MAX_ARRAY];
    char sbuf[MAX_ARRAY][MAX_STR_LEN];
    ub2 rlen[MAX_ARRAY]; /* return length */
    OCIDefine *defhdl;
} dltype;

extern int errno;

#define SQL_LEN 2048

#ifdef NULL
#define NULL 0
#endif

#ifdef NULLP
# define NULLP (void *)NULL
#endif /* NULLP */

#ifdef DISCARD
#define DISCARD (void)
#endif

#ifdef sword
# define sword int
#endif

#ifdef ub1
# define ub1 unsigned char
#endif

#define NA -1 /* ANSI SQL NULL */
#define VER7 2
#define NOT_SERIALIZABLE 8177 /* ORA-08177: transaction not serializable */

#define ADR(object) ((ub1 *)&(object))
#define SIZ(object) ((sword)sizeof(object))
#define SID(sid) ((sid == -1) ? 0 : sid)

/* For get_statement */

#define END_OF_FILE -1
#define COMMENT 1
#define SQL_STMT 2
#define SET_FETCHROW 3

#define OCIhalloc(envh,hndl,htyp) \
    if((status=OCIHandleAlloc((dvoid *)envh, (dvoid **)hndl,htyp,0,(dvoid **)0))!=OCI_SUCCESS) \
        sql_error(envh,status,0); \
    else \
        DISCARD 0

#define OCIhfree(hndl,htyp) \
    if((status=OCIHandleFree((dvoid *)hndl,htyp)) != OCI_SUCCESS) \
        fprintf(stderr, "Error freeing handle of type %d\n", htyp)

#define OCItaget(hndl,htyp,attp,size,atyp,errh) \
    if((status=OCIAttrGet((dvoid *)hndl,htyp, (dvoid *)attp, (dvoid *)size,atyp,errh)) != OCI_SUCCESS) \
        sql_error(errh,status,1); \
    else \
        DISCARD 0

#define OCIsaset(hndl,htyp,attp,size,atyp,errh) \
    if((status=OCIAttrSet((dvoid *)hndl,htyp, (dvoid *)attp,size,atyp,errh)) != OCI_SUCCESS) \
        sql_error(errh,status,1); \
    else \
        DISCARD 0

#define OCIsexec(svch,stmh,errh,iter) \
    if((status=OCIStmtExecute(svch,stmh,stmh,errh,iter,0,NULL,NULL,OCI_DEFAULT)) != OCI_SUCCESS) \
        sql_error(errh,status,1); \
    else \
        DISCARD 0

#define ISOTXT "alter session set isolation_level = serializable"
#define PDMLTXT "alter session force parallel dml parallel (degree %s)"
#define PDDLTX "alter session force parallel ddl parallel (degree %s)"

#endif /* QSTREAMPL_H */

tempitem.ctl
unrecoverable
load
append
into table temp_item
append
fields terminated by '|'
(
    l_shipdate date "yyyy-mm-dd",
    l_orderkey ,
    l_discount ,
    l_extendedprice ,
    l_suppkey ,
    l_quantity ,
    l_returnflag ,
    l_partkey ,
    l_linestatus ,
    l_tax ,
    l_commitdate date "yyyy-mm-dd",
    l_receiptdate date "yyyy-mm-dd",
    l_shipmode ,
    l_linenum ,
    l_shipinstruct ,
    l_comment ,
)

tshut
#!/bin/ksh

if [ "$1" = "abort" ]; then
    svrmgrl << !
    connect internal
    shutdown abort
    exit
    !
else
    svrmgrl << !
    connect internal
    shutdown immediate
    exit
    !
fi

```

```
tstart
#!/bin/ksh

svrmgrl << !
connect internal
shutdown immediate
startup pfile=$ORACLE_HOME/dbs/run_power.ora;
alter system switch logfile;
alter system switch logfile;
exit
!
```

Appendix F. Database Layouts

DB Entity	Volumes	Disks/Volume
Lineitem	sf1000_l_<1:24>	42
Orders	sf1000_o_<1:24>	42
Partsupp	sf1000_ps_<1:4>	42
Parts	sf1000_p_<1:4>	42
Supplier	sf1000_s_<1:4>	42
Customer	sf1000_c_<1:4>	42
Materialized Views	sf1000_ma_lineitem_1 sf1000_mj_locs_<1:44> sf1000_mj_ppssrn_<1:8>	230 230 230
Indexes, Constraints	sf1000_ind_lored_1_<1:8> sf1000_ind_oclokod_<1:4> sf1000_ind_okey_<1:4> sf1000_ind_pcbp_<1:4> sf1000_ind_ptp_<1:4>	230 230 230 230 230
sf1000_ij_1_lrid_<1:8>	230	230
	sf1000_ij_1_nullrid_1	230
	sf1000_ij_1_odate_<1:28>	230
	sf1000_ij_1_olid_<1:8>	230
	sf1000_ij_1_ppkey_<1:20>	230
	sf1000_ij_1_rdate_<1:16>	230
	sf1000_ij_1_sdate_<1:24>	230
	sf1000_ij_p_nnpsps_<1:4>	230
	sf1000_ij_p_ppsn_<1:4>	230
	sf1000_mj_1_log_1	230
	sf1000_mllog_1	230
	sf1000_molog_1	230
	sf1000_constr_<1:4>	230
	sf1000_maint_1	230
Default (for user tpcd)	sf1000_ts_default_<1:4>	230
Temp	sf1000_temp_<1:32>	230
Undo	sf1000_rbs_<1:4>	42
System (contains nation and region)	sf1000_sys_1	42
Log Files	sf1000_log_<3:4>	16 (LUNS)

The A5200 configuration was composed of 42 arrays of 22 disks each, for a total of 924 disks, utilized as follows:

- + For the volumes containing the tables, rollback segments, and system file, each volume was created as RAID 5 using 42 disks, 1 disk from each array.
- + For the volumes containing the indexes, materialized views, and temp, 920 of the disks were logically grouped into 4 sets of 230 disks each. The volumes were then created using the 4 sets in round-robin fashion.

The A3000 configuration was composed of 4 arrays of 35 disks each, utilized as follows:

- + For the log files, 4 LUNs of 5 disks were created as RAID 0 on each array. The volumes were created as RAID 0+1 using 8 LUNs from 2 arrays, mirrored by 8 LUNs from the other 2 arrays.


```

-----
SVRMGR=
SVRMGR= # set fetchrows 10;
SVRMGR=
SVRMGR= select * from lineitem
    2> where rownum <= 10
    3> order by l_orderkey;
L SHIPDAT L_ORDERKEY L_DISCOUNT L_EXTENDED L_SUPPKEY L_QUANTITY L_L_PARTKEY L
L_TAX L_COMMITD L_RECEIPT L_SHIPMODE L_LINENUMB L_SHIPINSTRUCT
L_COMMENT
-----
1997-04-3 619063108 .05 2850.2 7836493 2 N 67836492 O
.05 1997-03-0 1997-05-2 TRUCK 5 NONE My 5
344WO03m101ML0Kw03w1
1997-04-2 619063360 .05 31352.16 7416620 21 N 172416585 O
.08 1997-06-2 1997-05-1 TRUCK 12 DELIVER IN PERSON
600PPB1XNC4B3h4
1997-04-1 619063458 .06 56238.39 9874378 39 N 67374371 O
1997-04-2 1997-04-2 REG AIR 1 TAKE BACK RETURN
517NAXZi1h6i26
1997-04-3 619063495 .08 55131 1408244 46 N 153908198 O
02 1997-03-1 1997-05-1 TRUCK 3 NONE
Rz10z0K0NAlP yyByxmQ2g01h5AJ 026Bm1xk7zR2
1997-04-2 619063522 .03 8907.85 6024347 29 N 193524327 O
.05 1997-06-2 1997-05-0 TRUCK 6 COLLECT COD
BmMxm10zn01MyxxR05BgyQ0z0Pm14gmBx0J 3561hNj
1997-04-2 619064006 .03 96102.31 163997 47 N 65163984 O
08 1997-04-2 1997-05-1 SHIP 2 TAKE BACK RETURN
2w1LAQw5L7nP4A50R
1997-04-2 619064006 .03 13531.2 1071158 12 N 11071157 O
.01 1997-04-2 1997-05-1 AIR 3 NONE
252L5x1y7yL1yMx0z04 Cx1LjPn
1997-04-2 619064632 .05 37488.96 698575 24 N 10688574 O
.05 1997-02-1 1997-05-1 SHIP 5 DELIVER IN PERSON
Mx3Oyy0BgB1M2i11
1997-04-2 619064065 .07 7083.48 4054835 4 N 79054820 O
08 1997-05-0 1997-04-2 RAIL 2 NONE
R6xkhpPOCLWjhk2n
1997-04-2 619064134 .05 6084.35 1529214 5 N 106529193 O
Oxg14g4mMghz4ONBOy1n5N 0 A5Mj
10 rows selected.
SVRMGR=
SVRMGR= select * from orders
    2> where rownum <= 10
    3> order by o_orderkey;
O_ORDERDA O_ORDERKEY O_CUSTKEY O_ORDERPRIORITY O_SHIPPRIO O_CLERK
O_TOTALPRI O_COMMENT
-----
1997-10-2 348412177 5182135 4-NOT SPECIFIED 0 Clerk#000019743 O
10181.5 PmMwKcQgNzLR0A4Q0LM62Nj6B10x2h1k2KCQ K5 1BQ02gggKP
wPmBm6w51WS0M30A2FF0AJ406
1997-10-2 3484209539 7284577 2-HIGH 0 Clerk#000195256 O
158679.33 4Pzj64LMxNgjLSzN1SgMyg5gJBQ
1997-10-2 3484251233 5216564 1-URGENT 0 Clerk#000965699 O
252395.6 B42KwM0KwmQz01LRN xzhBj036mR0Cm hig35yOj1ghjxk
1997-10-2 3484284967 9488830 5-LOW 0 Clerk#0000011206 O
210050.09 m Lma2KL 00L0h7SwBxi3wklz0z0z0lnhwCRxP 61BPy 3Nj44i7K
1997-10-2 3484341510 120623242 5-LOW 0 Clerk#000787733 O
88991.678 5 xkAlB1zmnM71Pmz3R2P6R175y4B6hjBLCLWSc 3iz0miN14ij0SLnQn63j7y1PPkLN4wC
1997-10-3 3484394595 51167155 4-NOT SPECIFIED 0 Clerk#000658147 O
1997-10-9 9452Lx0h140COLx 22h4Nlnhw3jBlEx24Qj17hLwwh1Pm1xLLR2061Q6j
1997-10-2 3484468869 10321825 3-MEDIUM 0 Clerk#000733347 O
98441.53 gj 3h1lgRnOL2kmg0hxx04601hJhLMBSS056SLaxiQkN31F3OmhmRgM4LAl1
1997-10-2 3484489158 95914478 4-NOT SPECIFIED 0 Clerk#000331498 O
278714.82 Nh0Bx3mNSgPzPbnQhNxpAI0QOM2R2QR2
1997-10-2 3484567520 131890628 4-NOT SPECIFIED 0 Clerk#000562231 O
1997-10-6 45LkAkP163k6mjhhnm5K kw5P44wP2AML6hj0wnk
1997-10-2 3484628669 28811101 3-MEDIUM 0 Clerk#000682319 O
43670.06 4NLR01M7Qhnhzz201jgCgx03m0KMQkL1h26c0mANIL4MQ NB03hB
10 rows selected.
SVRMGR=
SVRMGR= select * from parts
    2> where rownum <= 10
    3> order by p_partkey;
P_PARTKEY P_TYPE P_SIZE P_BRAND P_NAME
P_CONTAINE P_MFGR P_RETAILPR P_COMMENT
-----
164076789 ECONOMY POLISHED BRASS 37 Brand#32 thistle almond pink
chiffon light SM CAN Manufacturer#3
1757.58 Sj7CC2C
164076797 PROMO BURNISHED NICKEL 37 Brand#21 green beige cornsilk
flora JESSE WRAP BAG Manufacturer#2 1765.59
Niy7Qh157j 4zRIA2B1
164076966 ECONOMY POLISHED COPPER 37 Brand#25 burnished dark red puff
chocolate WRAP DRUM Manufacturer#2 1934.76
164076797 1x1701OCxgMy
164077065 LARGE BRUSHED STEEL 37 Brand#51 lace magenta black
olive SM JAR Manufacturer#5
1033.86 1C6hw50 mzQL
164077130 PROMO BURNISHED NICKEL 37 Brand#43 linen magenta pink
white JVOY JUMBO PKG Manufacturer#4
1098.93 nck1k70z5MnKcYn
164077137 SMALL BURNISHED BRASS 37 Brand#13 forest floral navajo
blush bisque WRAP DRUM Manufacturer#1 1105.93
1M1K6Kx85Kak13n3AAA0
164077140 LARGE PLATED TIN 37 Brand#44 plum dark slate blue
violet JUMBO PACK Manufacturer#4 1108.94
5wZrh2
164077142 SMALL BURNISHED BRASS 37 Brand#15 mint cornflower
honeydew JUMBO DRUM Manufacturer#1
1110.94 hhg xCh1h
164077254 SMALL BRUSHED STEEL 37 Brand#42 purple black sky bisque
azure 104wM Manufacturer#4 1223.05
164077265 SMALL ANODIZED COPPER 37 Brand#44 misty plum almond
agumarine floral LG PACK Manufacturer#4
1234.06 xj617h
10 rows selected.
SVRMGR=
SVRMGR= select * from partsupp
    2> where rownum <= 10
    3> order by ps_partkey, ps_suppkey;
PS_PARTKEY PS_SUPPKEY PS_SUPPLYC PS_AVAILQT PS_COMMENT
-----
152926188 426234 355.36 14817 SgBBrQyLOn1xM0nRl2R
542NkN0Ojzmh4gi0PLC15k7401j6iLM1h3vi1050110 1zyFm17PmK1330

```

```

152926188      7926215      245.2      9913 BR40RJ7y5yQ2Y7mhyLNgw2zkim1N3wQOOPQ
PCR K 0L64N6Gycmj1lIONNgjRzLS z7QMB16Pcn M52
152926189      426235      689.36      7017 MNkw B12R3 y0k
CN1iHmQMBi4g4n3y1MB20AMPXA14SN5yw3NC4S143SR6g7BgkkihSB42SiB7m
152926189      2926190      663.74      9402 n0BkygAzCyRl0x0BxiBoxy2x107 m
yP1n00SCBN190568813gg0lPkPzszMq1PCQM30l4whzRkXSj
152926189      2426205      819.94      8022
6LOj1j47w61l65R6xCwy0wn0h60CxxAOBS07AOAiMCR0NNw0Ch CAL1j5QlN0Q3MR27nPNg 64yC
152926189      7262220      626.94      6136 R06g6hlyN55 MB7Bzmi4N5gN M20LQ0y5j
6KRPA5ANg6y7LoY0bJ2x0XMC5xga0A3mmmb1ChN1NR6BBWw
152926190      426236      39.81      17 734QxPmyNQMHa10k1Mh5Q2A15331 hzz
71AOxyR7M27B2BzQw4SmkzC7w3y5s14MA4Annk5W09n6wC
152926190      2926191      873.23      1929 SC6L1K2CixwCA3m3zn
zSwBgMC61nABmmxQ6M2S1g7yLmzMR5y15CLANQ0M9Y7 Lg231lin 70A60wmx
152926190      5426206      224.86      8329
C2znNOB26R6kzwyjKgL5ZLbY20BAAB57x1NPK1C2AyP42wAMRGMyz2AMNg2n2glgZLNmNkNWAM1AR5ixQA
152926190      7926221      311.75      7507 PC1
y1y0yR7M27B2BzQw4SmkzC7w3y5s14MA4Annk5W09n6wC
10 rows selected.
SVRMGR>
SVRMGR> select * from supplier
2 where rownum <= 10
3 order by s_supplier;
S_SUPKEY S_NATIONKE S_COMMENT
S_NAME S_ADDRESS S_PHONE
S_ACCTBAL
-----
102488      8
SnaWQpQ0nXcZ6wi3SkLk155g5j0xZxSkj15ArvN174zx545AmikxQlxRazmBzBgX
Supplier#000102488 R36SS61gMAZ2Alln
3587.6 18-406-625-7148
102489      7
6OM2CN3A8X0L2P3AzK6LXnn155843k1m6x5kB23411Mm411Nnm04P5316CgSwm1Ax70BkPh g3Rgh
Supplier#000102490 Np01314jy0ORRg1hw55z2g2NNPpy0XN 17-373-811-5499
3750.3
102490      8
N9h1350PmCMS
zRAK32Q65PvCyQokCBkQR2C35425wKMAmNwlnhw70525RGp0j9kPvBQXNvgh6Ax00LR
Supplier#000102490 n14j2B5w5kPgPAL6L15MNA14P RojBmLCf. 18-902-692-4703
-643.21
102491      4
k Svkl LzxMqQ0677M7R1gn3Ch1h1MigmQ0P32R63kC
nhg5j47R0RCQPSyEw0g3ZgAMmPbP1nF Supplier#000102491 FyLK031/mhgM
14-658-680-6473 439.26
102492      16
xxkQbny5ml0R00jMj2N107x2152NW0yJwmw
Supplier#000102492 3lAxjy772SO BPljhynM7MB11i27 y0xww7A7M 26-299-727-6944
-470.11
102493      13
nh207NRgkR0m3w7C3j6MlnCnh Supplier#000102493
173kkkwm1mL2iR32j6l77362z0H7zgQR0I1MR R0 M1ml0kY5Lz1B6n Nmgixz4 23-591-714-4065 2182.92
102494      15
RRBN3wwQinQkCx5BwP4M3QBxN3
Supplier#000102494 51nmj1yNk31R4mm 25-441-261-2793
1316.89
102495      13
kNRNXP jg4BjngPOBOIPIB0mhl4w1xg7SgQ 23zMA
NOC31kmy105gN4A5597C0JLLSnmS1mC10k1g Supplier#000102495 zw03Q 1wL7Pz
23-474-324-5024 -86.06
102496      8
w130k7yNGNgrNL3R6g3hSAXK5K2y0h2j vC5xLNQ0404R3x
lyLPh761mm0hx5Q1xSPnQj0xON Supplier#000102496 MCN2zP3L64ni
18-456-274-1525 -929.23
102497      10
7xykRGtBetter Business Bureau 42Lkg5
OCL5Y1C1g1g7POPOC0h1L1j1mJW0YLS3Recomm Supplier#000102497
hlh511C1g73j1nnORABL1znQ41Mm hkhmyj64 20-860-439-2933 346.69
10 rows selected.
SVRMGR>
SVRMGR> select * from customer
2 where rownum <= 10
3 order by c_custkey;
C_CUSTKEY C_MKTSEGME C_NATIONKE C_NAME C_ADDRESS
C_PHONE C_ACCTBAL C_COMMENT
-----
92819276 BUILDING 10 Customer#092819276
6R7N025k5kmMSxyP0RHyLXwywNk 20-229-726-8069 3932.73
10282030 BUILDING 10 Customer#092820300 57BA716MzC13x7xjkyN
32-814-168-2653 9651.86 5M26k6h6A0Snm2CM45nJLCwNMkm5OS85j1zlm43CmbmH0LWLLP
92820411 BUILDING 10 Customer#092820411 7R15k3002wynhojxj7n
30-698-261-9486 415.58 kyvNj0ZwLl71Cj0jgmzggZnLnwyLry6gwn02gS LAS5PM
4Mmyx4X5mmNRSyQ1xQmb31B6w1RQ3
92820522 BUILDING 19 Customer#092820522 xch45L7 PoyAj771
29-843-646-9092 7535.19
1hC3CM3gMQR04KQBL67X5nN1h5nRwAAn0hCWPN71wPOCmA0mPw0XRRP04x h633hS56NWPPAMBNwz
92822571 BUILDING 13 Customer#092822571
Nky1R1RMJ19C4P1Q0RQ2C11j0 i3mw44OB8 0-305-154-2255 6196.16 kzLLP
1i1j11LA5x2RRMBzMBMg10jyWknywn55mxC g5C171 AAnC
92823443 BUILDING 13 Customer#092823443 06mCAPR1Pq4BMAN336h
71y1y10mPvC7L2 23-553-422-8813 19-863 1A4y6yAL6j5xABXj 1yhQ3j445hS0S0w
4zEQ0P01N1NBA04KwzznR10mz6AmLh5H0PRA 1yS72k1
92824087 BUILDING 0 Customer#092824087 gnizAL 343
10-332-986-7235 4057.02 kKpJRS4NFjKyBgg05hAhPOSh5yLwy6Ql3hPNOPrhwlxy21Rma
Q731R1RMJ19C4P1Q0RQ2C11j0 xPnL5 xlln40
92824151 BUILDING 10 Customer#092824151 xPyw43hRLRB15w
20-505-964-5488 8064.39 kymzLCrCnOmCh604R16hB7w1AyA1516z0NRS
56Rk5x1LLN14LL2LSOCXNPgyA
92824224 BUILDING 5 Customer#092824224 5Qk63zwj0NMnlh417g3
imx2011B5043gN02 15-617-677-777 5861.18
kg0N0Qj6x0P4A41gNALPS1C2j0jhm7wM55y7Mzkb
92824819 BUILDING 3 Customer#092824819 P04C0h0AQOS0PmkyAnA4k0
13-754-412-5048 8675.86 M5LxwJ6MgwMbz5j1PIQAhCBL0wn45kxPkzRm6RR770NAP 067z2C1LP
10 rows selected.
SVRMGR>
SVRMGR> select * from nation
2 where rownum <= 10
3 order by n_nationkey;
N_NATIONKE N_NAME N_REGIONKE N_COMMENT
-----
0 ALGERIA 0 2Cxlh17
L1iik6hMh30i1zn932CwPfcikyLk6hMzSRA
1 ARGENTINA 1
Zqn30kzw1wLn7L73H0hCgn56kP5PyRikg1B7L1
2 BRAZIL 2
gLmS0nACAPn5032x PnL71RCNPgPxmC0jNg4k OiAg57COS0m1NwCnOyLx40R SC y20gPPAKNk5ShxRhR5
1 rHb0 047R26 2B1M 1 4yMo AhnQ0L5
3 CANADA 1
wQAM662AwiByC17CxmzRwNR5nA104 x 4 11m15126 Cxj
4 EGYPT NMQmLx0k1n10j2m3AhY4NR1QQ1LS07j2QSlyN 0
5 ETHIOPIA 0 NS7n LSOP Oz5n1AlB2S02nN01Mh4SBxP
rHb0 047R26 2B1M 1 3mjmi2l S
6 FRANCE 3
3Lk3zhNnh1P4w370xRxyN15wn 2 3nO4Rkwo CmzBB 516mAg
7 GERMANY 2
1Byw4OM3QyNPA 3
8 INDIA 2
M1ly1N1Mmb0j7PL4C kxxk0gPQ713w6B7R2QK0040x14Q21w76JRL71 2
9 INDONESIA 2 S1PmQ07L1j
7ABj6mC1AQk3nLw173BPzxCwjZm4zLZgg6nno2j0w xzC669PkykRPgM
10 rows selected.
SVRMGR>
SVRMGR> select * from region
2 order by r_regionkey;
R_REGIONKE R_NAME R_COMMENT
-----

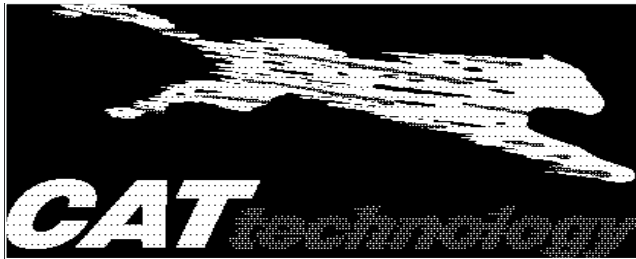
```

```

-----
0  AFRICA
xSx31zz31C11z40Aimm05AjiOxC3AMMNOgC0kACgwnngg3glP7LLLywlQy7R
1  AMERICA
kgyh3LSnC72k6z1Az0LP3k2L4QB1QL106730j01SPj0ngQ7CO100SBgmgRQ4lgPCmk21A425iklyAR4y
2  ASIA
RjySB3RLwkyPkWMM2R1BQ xAzkOgkjm1l0gAgh1 NSg6x1M1A1lzm6mOR0Ajd nhRA77NgRxBwLlM6Py
3  EUROPE
z1SL7Qwg12hMBL51h1z0M45QkjShwSyiO04MLOh7wn1ARLQPyPayAii157611Li7AlnR1S RQ4SLny7B
4  MIDDLE EAST
R1lxmhPLz3Cy2mNlg4QMBnNASM ACKi MPki70i
5 rows selected.
SVRMGR>
SVRMGR> spool off;

```

Appendix H. Programs and Scripts



Sales Quotation

Quote Number: KHRQ0121SMI
 Date: 2/10/99
 FOB: Warehouse
 Terms: Net 30, 1.5% per
 month after 30 days
Quote Valid for 60 Days

Brad Carlile
Manager, Strategic Apps. Engineering
Sun Microsystems, Inc.
8300 SW Creekside Place
Beaverton, OR 97008
Ph: 503-520-7622

CAT Technology, Inc.
Kandis Rodda
1900 Camden Ave., Ste 201
San Jose, CA 95124
Ph: 408-369-7878
Fax: 408-369-7870

ITEM	PART NO.	DESCRIPTION	QTY	UNIT PRICE	AMOUNT
1	E10000-3	ULTRA E10000 BASE CABINET	1	\$ 165,000.00	\$ 165,000.00
2	X2760A	E10000 SYSTEM BOARD EMPTY	16	\$ 48,750.00	\$ 780,000.00
3	X2570A	CPU, 400MHz, 4MB	64	\$ 12,750.00	\$ 816,000.00
4	X7025A	E10000 MEM BOARD, UNPOPULATED	15	\$ 7,500.00	\$ 112,500.00
5	X7023A	OTP MEMORY 1GB (8*128MB)	60	\$ 7,125.00	\$ 427,500.00
6	X2730A	E10000 I/O BOARD	16	\$ 5,625.00	\$ 90,000.00
7	X2750A	E10000 PCI MODULE	1	\$ 7,500.00	\$ 7,500.00
8	X3875A	E10000 AC INPUT MODULE	4	\$ 2,250.00	\$ 9,000.00
9	X9685A	E10000 POWER SUPPLY	8	\$ 3,000.00	\$ 24,000.00
10	X9681A	E10000 POWER CONTROL	1	\$ 750.00	\$ 750.00
11	X9671A	E10000 FAN TRAY	16	\$ 1,500.00	\$ 24,000.00
12	X2720A	E10000 CONTROL BOARD	1	\$ 15,000.00	\$ 15,000.00
13	X3850A	E10000 POWER CORD, FOR N.A.	4	\$ -	\$ -
14	X979A	OPT CABLE DSCSI 12M	2	\$ 198.75	\$ 397.50
15	X1062A	OPT SBUS F/W DWIS/S ADAPTER	2	\$ 971.25	\$ 1,942.50
16	X6730A	FCAL 100MB/S SBUS HOST ADAPTER	42	\$ 2,025.00	\$ 85,050.00
17	X1065A	OPT ULTRA DWIS/S HOST ADAPTER	8	\$ 971.25	\$ 7,770.00
18	X6534A	A3000 15*9.1GB/7200 FWSCSI	4	\$ 72,937.50	\$ 291,750.00
19	X6516A	RSM 9.1GB/7200 FWSCSI	80	\$ 1,312.50	\$ 105,000.00
20	X3800A	OPT PWR CORD FOR STOREDGE	4	\$ -	\$ -
21	SG-XARY523A-1200G	1200-GB SUN STOREDGE A5200	7	\$ 445,237.50	\$ 3,116,662.50
22	6515AR4	RSM 7*9.1GB/7200 FWSCSI	2	\$ 13,950.00	\$ 27,900.00

TOTAL	\$ 6,107,722.50
--------------	------------------------