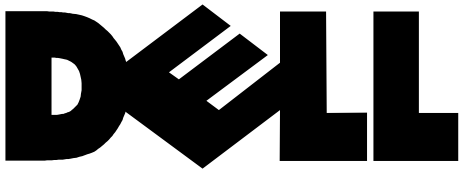
Dell | POWER
PowerApp | PowerEdge | PowerVault

TPC Benchmark ™ R
Full Disclosure Report
DELL PowerEdge 6600/1.6GHz
Using Oracle9i Database Enterprise Edition Release 2 with Partitioning



First Edition

Submitted for Review

October 11, 2002

Dell Computer Corporation PowerEdge 6600 Server with Oracle9i Database Enterprise Edition Release 2 with Partitioning on Red Hat Linux Advanced Server 2.1

First Printing October 2002

Printed in U.S.A.

DELL Computer Corporation believes that the technical, pricing and discounting information in this document is accurate as of its publication date. 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. DELL Computer Corporation does not warrant or represent that a user can or will achieve similar performance as expressed in this document.

THE TERMS AND CONDITIONS GOVERNING THE SALE OF DELL HARDWARE PRODUCTS AND THE LICENSING OF DELL SOFTWARE CONSIST SOLELY OF THOSE SET FORTH IN THE WRITTEN CONTRACTS BETWEEN DELL AND ITS CUSTOMERS. NO REPRESENTATION OR OTHER AFFIRMATION OF FACT CONTAINED IN THIS DOCUMENT INCLUDING BUT NOT LIMITED TO STATEMENTS REGARDING PRICE, CAPACITY, RESPONSE-TIME PERFORMANCE, SUITABILITY FOR USE, OR PERFORMANCE OF PRODUCTS DESCRIBED HEREIN SHALL BE DEEMED TO BE A WARRANTY BY DELL FOR ANY PURPOSE, OR GIVE RISES TO ANY LIABILITY OF DELL WHATSOEVER.

DELL assumes no responsibility for any errors that may appear in this document. DELL reserves the right to make changes in specifications and other information contained in this document without prior notice, and the reader should in all cases consult DELL to determine whether any such changes have been made.

Abstract

This report documents the methodology and results of the TPC Benchmark R test conducted on the PowerEdge 6600 Server using Oracle9i DB Enterprise Edition Release 2 with partitioning in conformance with the requirements of the TPC-R Benchmark Specification. The operating system used for the benchmark was Red Hat Linux Advanced Server 2.1. The application was written in C and compiled using the GNU compiler.

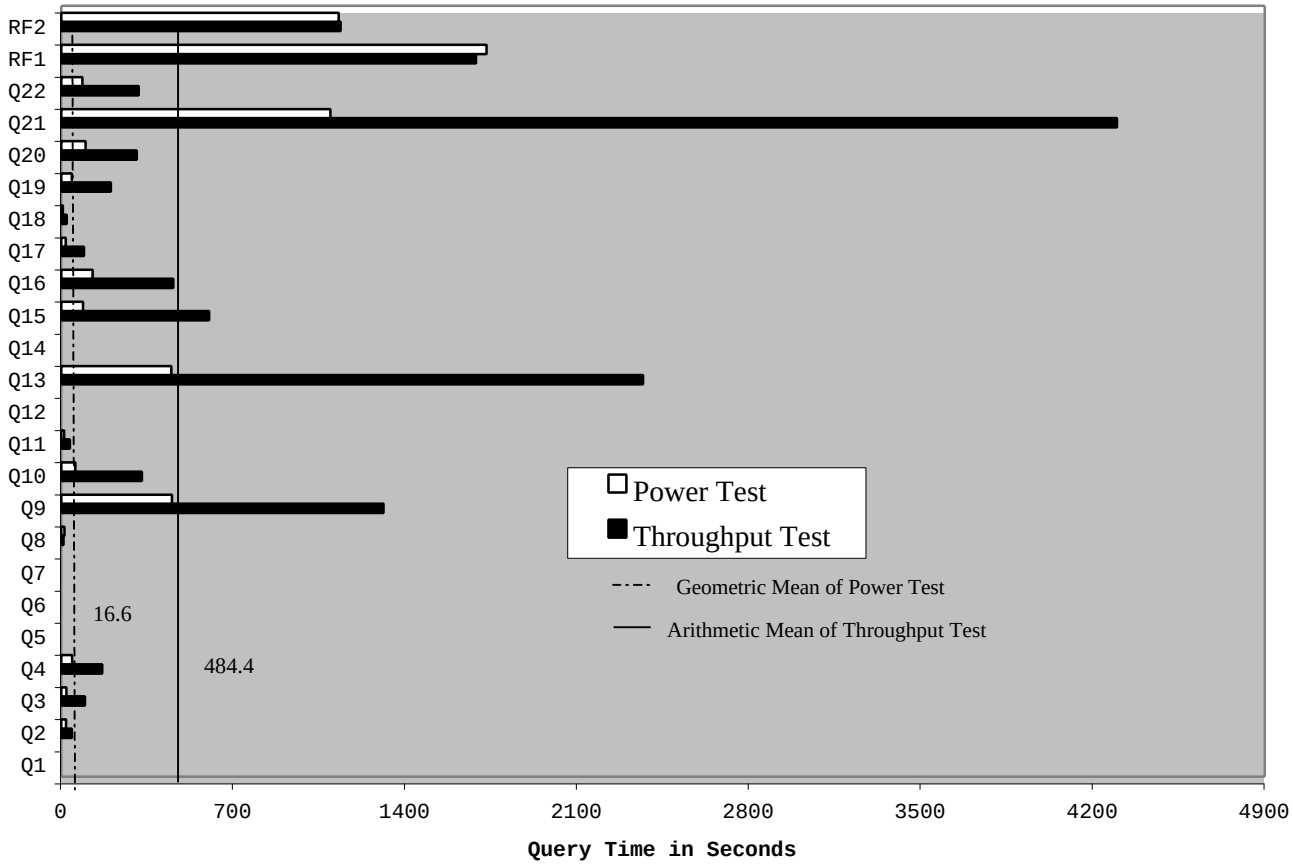
Hardware	Software	Total System Cost
Dell PowerEdge 6600 with Quad 1.6GHz Intel Xeon MP Dell PowerVault 22X Storage Enclosures	Oracle9i Database Enterprise Edition Release 2 with Partitioning Red Hat Linux Advanced Server 2.1	$154,726

TPC-R Power@100GB	TPC-R Throughput@100GB	QphR@100GB	$/QphR@100GB
12802.6	1548.5	4452.5	$35.00

The Transaction Processing Performance Council (TPC) developed the TPC-R Benchmark. The TPC was founded to define transactions processing benchmarks and to disseminate objective, verifiable performance data to the industry.

In order to verify compliance to the TPC-R benchmark specification, Lorna Livingtree, Performance Metrics, Inc., audited the benchmark configuration, environment and methodology used to produce and validate the test results, and the pricing model used to calculate the price/performance. The auditor's letter of attestation is attached as Appendix I.

Dell POWER PowerApp | PowerEdge | PowerVault	PowerEdge 6600/1.6GHz Using Oracle9i Database Enterprise Edition Release 2 with Partitioning	TPC-R Revision 2.0 Report Date: October 11, 2002
Total System Cost **$154,726**	**TPC-R Composite Query per Hour** **4452.5 QphR@100GB**	**Price/Performance** **$35.00/QphR@100GB**
Database Size 100GB	**Database Manager** Oracle9i Database Enterprise Edition Release 2 with Partitioning	**Operating System** Red Hat Linux Advanced Server 2.1
Other Software None		**Availability Date** **April 4, 2003**



Database Load Time	Total Disk/Database Size	Load Includes Backup
14:08:14	29.84	N
RAID (Base tables only)	**RAID (Base tables and auxiliary data structures)**	**RAID (Everything)**
N	N	Y

System Configuration

Processors: 4 – 1.6GHz/256kB L2/1MB L3 Intel Xeon MP
Memory: 4GB RAM
Disk Drives: 2 – 18GB 10k HDDs
4 – 36GB 10k HDDs
84 – 36GB 15k HDDs

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

DELL	PowerEdge 6600/1.6GHz using Oracle9i Database Enterprise Edition Release 2 with Partitioning	TPC-R Revision 2.0.0
		Report Date 10/11/02
Price/Performance	Performance	Availability Date
$35.00 / QphR@100GB	4452.5 QphR@100GB	4/4/2003

Description	Part No.	Third Party Brand	Third Party Price	Unit Price	Quantity	Extended Price	3 Yr Maintenance Price
Database Server Hardware:							
SYSTEM: PowerEdge 6600 1.6GHz/1MB L3	220-8304	1	1	$7,158	1	$7,158	
PROC: Quad Procs, 1.6GHz/1MB Xeon	311-1528	1	1	$14,999	1	$14,999	
MEM: 4GB (4x1GB)	311-1549	1	1	$2,599	1	$2,599	
DISK: 1x18GB 10k SCSI	340-3767	1	1	$249	2	$498	
DISK: 4x36GB 10k SCSI	340-3784	1	1	$996	1	$996	
NIC: Dual Integrated Broadcom Gigabit NIC Adapters	Included in SYSTEM.					$0	
CONTROLLER: LSI MegaRAID Elite 1650	1930472	3	3	$979	7	$6,853	
SPARES: LSI MegaRAID Elite 1650	Spares priced to spec.	3	3	$979	2		$1,958
ACCESSORY: Rapid Rail Racks	310-1436	1	1	$349	1	$349	
MAINT: 3 Years 4Hour-7x24, SILVER Support (Initial/2Yr Ext.)	900-6370/900-6352	1	1	$1,783	1		$1,783
POWER: AC Switch (Dual Power Cords)	310-1446	1	1	$299	1	$299	
MONITOR: DELL E551 15"	320-0960	1	1	$149	1	$149	
Database Server Software:				**Database Server Hardware Subtotal:**		**$33,900**	**$3,741**
OS: Red Hat Linux Advanced Server 2.1	420-1069*	1	1	$2,499	1	$2,499	
MAINT: PESS Gold for Advanced Software Support, 3 years	950-0117*	1	1	$300	1		$300
MAINT: PESS Gold 1st year Premium Services	950-0128*	1	1	$1,151	1		$1,151
MAINT: PESS Gold years 2 & 3 Premium Services	950-0119*	1	1	$666	1		$666
DBMS: Oracle9i Database Enterprise Edition Release 2, Named User Plus for 3 years	** Please see below.	2	2	$10,000	4	$40,000	
DBMS: Oracle Partitioning, Named User Plus for 3 years	** Please see below.	2	2	$2,500	4	$10,000	
DBMS MAINT: Oracle Database Server Support Package for 3 years	** Please see below.	2	2	$2,000	1		$6,000
Storage Hardware:				**Database Server Software Subtotal:**		**$52,499**	**$8,117**
SYSTEM: PowerVault 220S - 12 HDD Storage System	220-4476	1	1	$1,256	6	$7,536	
ACCESSORY: PowerVault 220 Locking Bezel	313-1308	1	1	$39	6	$234	
CTRL: Ultra3 SCSI Redundant Enclosure Management Module	340-3090	1	1	$634	6	$3,804	
POWER: Redundant Power Option	310-0677/310-0682	1	1	$299	6	$1,794	
ACCESSORY: Rapid Rails	310-0684	1	1	$114	6	$684	
DISK: 7x36GB 15k SCSI	340-3555	1	1	$4,893	6	$29,358	
DISK: 36GB, 15k SCSI	340-3088	1	1	$599	42	$25,158	
DOCUMENTATION: Storage Documentation Hard Copy	310-1321	1	1	$49	6	$294	
CBL: 2 Cables,PERC,12M, PV220S	310-1686	1	1	$99	6	$594	
MAINT: 3 Years 4Hr/7x24, SILVER Support (Initial/2Yr Ext.)	900-6790/900-6792	1	1	$1,099	6		$6,594
RACK: PE4210,DOORS,GRND,42U,PS	220-4493	1	1	$1,079	1	$1,079	
RACK ACCESSORY: 42U Rack, Interconnect Kit, PS to TS	310-1283	1	1	$119	1	$119	
RACK POWER: Power Strip, 6 Socket	310-0081	1	1	$69	1	$69	
RACK MAINT: 3 Years BRONZE Support (Initial/2Yr Ext.)	900-6220/900-6222	1	1	$283	1		$283
				Storage Subtotal:		**$70,723**	**$6,877**
				Total:		**$157,122**	**$18,735**
				DBMS Discount: Oracle Mandatory E-Business Discount (license and support)**		**($5,600)**	
				Dell Discounts:***		**($15,531)**	

3 Year Cost of Ownership	**$154,726**
QphR@100GB	**4452.5**
$ / QphR@100GB	**$35.00**

*** Red Hat pricing contact: Anthony Kotecki, tony_kotecki@dell.com, 512-728-1207. (See Appendix H)**

**** Oracle pricing contact: MaryBeth Pierantoni, mary.beth.pierantoni@oracle.com, 650-506-2118. (See Appendix H)**

***** Dell discount is based upon total system cost as purchased by a regular customer. Pricing may be verified by calling 1-800-BUY-DELL and referencing quote number 91268948 as the complex quote.**

Independently Audited by:	Lorna Livingtree	Performance Metrics

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

Dell POWER PowerApp | PowerEdge | PowerVault	PowerEdge 6600/1.6GHz Using Oracle9i Database Enterprise Edition Release 2 with Partitioning	TPC-R Revision 2.0 Report Date: October 11, 2002

Numerical Quantities Summary

Measurement Results:

Database Scale Factor	100
Total Data Storage/Database Size	29.84
Start of Database Load	9/29/2002 14:55:24
End of Database Load	9/30/2002 05:03:38
Database Load Time	14:08:14
Query Streams for Throughput Test	5
TPC-R Power	12802.6
TPC-R Throughput	1548.5
TPC-R Composite Query-per-Hour Metric (QphR@100GB)	4452.5
Total System Price Over 3 Years	$154,726
TPC-R Price Performance Metric ($/QphR@100GB)	$35.00

Measurement Interval:

Measurement Interval in Throughput Test (Ts) = 25573 seconds

Duration of stream execution:

	Seed	Start Date/Time	End Date/Time	Duration
Stream00	930050338	9/30/2002 14:20:06	9/30/2002 15:51:37	1:31:31
Stream01	930050339	9/30/2002 15:52:09	9/30/2002 19:02:51	3:10:42
Stream02	930050340	9/30/2002 15:52:09	9/30/2002 18:47:04	2:54:55
Stream03	930050341	9/30/2002 15:52:09	9/30/2002 18:39:00	2:46:51
Stream04	930050342	9/30/2002 15:52:09	9/30/2002 18:43:34	2:51:25
Stream05	930050343	9/30/2002 15:52:09	9/30/2002 18:56:20	3:04:11
Refresh		9/30/2002 15:52:09	9/30/2002 22:58:22	7:06:13

Numerical Quantities Summary

Timing Intervals in Seconds:

Query	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
Stream00	0.1	19.2	20.3	43.8	0.2	0.1	0.3	12.4
Stream01	0.1	51.0	122.7	46.6	0.2	0.1	0.5	7.2
Stream02	0.1	56.4	43.8	260.6	0.2	0.6	0.6	7.4
Stream03	0.1	30.2	119.7	113.3	0.1	0.1	0.4	10.9
Stream04	0.2	38.4	86.8	219.3	6.4	0.1	0.5	9.3
Stream05	0.1	41.1	106.2	196.5	0.2	0.1	0.6	11.5
Min Qi	0.1	30.2	43.8	46.6	0.1	0.1	0.4	7.2
Max Qi	0.2	56.4	122.7	260.6	6.4	0.6	0.6	11.5
Avg Qi	0.1	43.4	95.8	167.3	1.4	0.2	0.5	9.3

Query	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16
Stream00	451.1	57.9	11.7	0.1	448.6	0.1	88.3	127.3
Stream01	1070.1	447.3	45.0	0.1	2362.1	0.1	638.1	494.2
Stream02	1604.7	257.2	27.4	0.1	2642.8	0.1	803.0	410.3
Stream03	1689.0	286.0	31.6	0.1	2266.5	0.1	532.2	448.2
Stream04	1580.0	358.5	35.5	0.1	2342.8	0.1	514.6	467.5
Stream05	617.0	294.7	38.7	0.1	2226.8	1.7	522.3	457.9
Min Qi	617.0	257.2	27.4	0.1	2226.8	0.1	514.6	410.3
Max Qi	1689.0	447.3	45.0	0.1	2642.8	1.7	803.0	494.2
Avg Qi	1312.2	328.7	35.6	0.1	2368.2	0.4	602.0	455.6

Query	Q17	Q18	Q19	Q20	Q21	Q22	RF1	RF2
Stream00	18.7	7.0	43.4	98.8	1095.5	86.0	1731.0	1129.0
Stream01	62.2	35.5	232.6	285.8	5453.0	88.0	1680.0	1138.0
Stream02	42.5	15.2	309.7	292.1	3372.7	347.3	1658.0	1150.0
Stream03	93.7	48.3	123.2	395.9	3514.0	307.8	1676.0	1126.0
Stream04	138.6	10.5	170.9	272.3	3599.2	434.1	1739.0	1146.0
Stream05	126.0	1.5	172.6	288.5	5549.0	397.5	1689.0	1128.0
Min Qi	42.5	1.5	123.2	272.3	3372.7	88.0	1658.0	1126.0
Max Qi	138.6	48.3	309.7	395.9	5549.0	434.1	1739.0	1150.0
Avg Qi	92.6	22.2	201.8	306.9	4297.6	314.9	1688.4	1137.6

Table of Contents

General Items

Test Sponsor

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

DELL Computer Corporation and Oracle Corporation are the sponsors of this TPC Benchmark™ R result.

Parameter Settings

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

- *Database Tuning Options*
- *Optimizer/Query execution options*
- *Query processing tool/language configuration parameters*
- *Recovery/commit options*
- *Consistency/locking options*
- *Operating system and configuration parameter*
- *Configuration parameters and options for any other software in the pricing structure*
- *Compiler optimization options*

Providing a full list of all parameters and options can satisfy this requirement, as long as all those that have been modified from their default values have been clearly identified and these parameters and options are only set once.

***Comment** 1: In the event that some parameters and options are set multiple times, it must be easily discernible by an interested reader when the parameter or option was modified and what new value it received each time.*

***Comment** 2: This requirement can be satisfied by providing a full list of all parameters and options, as long as all those that have been modified from their default values have been clearly identified and these parameters and options are only set once.*

Details of system and database configurations and parameters are provided in Appendix A.

Configuration Items

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

- *Number and type of processors*
- *Size of allocated memory, and any specific mapping/partitioning of memory in the test.*
- *Number and type of disk units (and controllers, if applicable).*
- *Number of channels or bus connections to disk units, including their protocol type.*
- *Number of LAN (e.g. Ethernet) Connections, including routers, workstations, terminals, etc., that were physically used in the test or are incorporated into the pricing structure.*

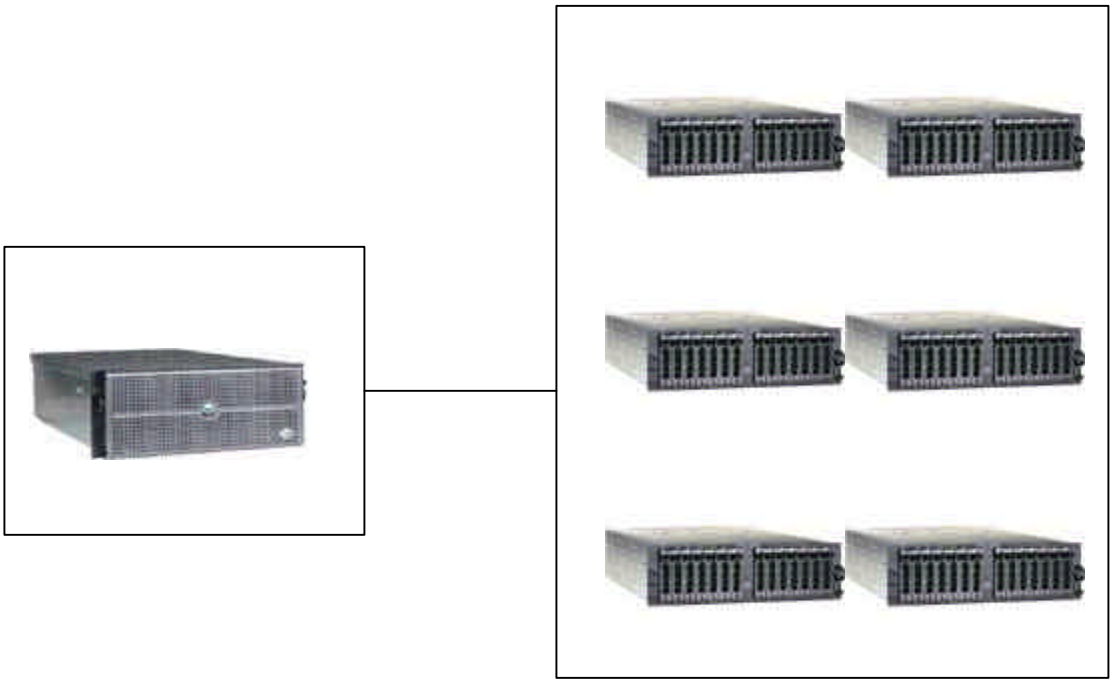
· *Type and the run-time execution location of software components (e.g., DBMS, query processing tools/languages, middle-ware components, software drivers, etc.).*

The System Under Test (SUT), a DELL PowerEdge 6600 Server, depicted in the next diagram, consists of:

- 4 - 1.6GHz Intel ® Xeon MP Processors, each with 1MB of L3 cache.
- 4GB RAM
- 7 - LSI MegaRaid 1650 RAID Controllers, each with 64MB RAM of cache.
- 6 - PowerVault 22X.
- 84 – 36GB 15k HDD
- 4 – 36GB 10k HDD
- 2 – 18GB 10k HDD
- Oracle9i Database Enterprise Edition, Release 2 with Partitioning

Measured and Priced Configuration

The measured and priced configurations are the same.



Processors:	4 – Intel Xeon MP1.6GHz/ 256kB L2/ 1MB L3
Memory:	4GB RAM
Controllers:	7 – LSI MegaRaid 1650 128MB Cache
Disks:	6 – PowerVault 22X
	84 – 36GB 15k HDD
	4 – 36GB 10k HDD
	2 – 18GB 10k HDD
DBMS:	Oracle9i Database Enterprise Edition, Release 2 with Partitioning

Clause 1: Logical Database Design

Table Definitions

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

Appendix B contains the scripts that create and setup the tables and indexes for the TPC-R test and qualification database.

Physical Organization of Database

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.
***Comment**: The concept of physical organization includes, but is not limited to: record clustering (i.e., rows from different logical tables are co-located on the same physical data page), index clustering (i.e., rows and leaf nodes ofan index to these rows are co-located on the same physical data page), and partial fill-factors (i.e., physical datapages are left partially empty even though additional rows are available to fill them).*

Appendix B contains the details of the physical organization of the tables and indices. The column ordering of tables LINEITEM, ORDERS, PARTS, SUPPLIER, PARTSUPP and CUSTOMER differs from that specified in Clause 1.4, and is disclosed in load100g_data.dat.

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 LINEITEM, ORDERS, PARTS, SUPPLIER, PARTSUPP and CUSTOMER, as well as some indexes and some materialized views. The details of the partitioning are disclosed in the syntax of the table and index definition statements in Appendix B. The same partitioning was used for the qualification database.

Replication

Any replication of physical objects must be disclosed and must conform to the requirements of Clause 1.5.5.

No replication was used.

Clause 2: Queries and Update Functions

Query Language

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

SQL was the query language used to implement all queries.

Random Number Generation

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

Version 1.3.0 of DBGEN and version 1.3.0 of QGEN were used to generate the random numbers for this TPC-R benchmark.

Substitution Parameters Generation

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

QGEN version 1.3.0 was used to generate the substitution parameters.

Query Text and Output Data from Database

The executable query text used for query validation must be disclosed along with the corresponding output data generated during the execution of the query text against the qualification database. If minor modifications (see Clause 2.2.3) have been applied to any functional query definitions or approved variants in order to obtain executable query text, these modifications must be disclosed and justified. The justification for a particular minor query modification can apply collectively to all queries for which it has been used. The output data for the power and throughput tests must be made available electronically upon request.
***Comment**: For query output of more than 10 rows, only the first 10 need to be disclosed in the FDR. The remaining rows must be made available upon request.*

Appendix D contains the query text and query output.

Query Substitution Parameters and Seeds Used

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

Appendix E contains the seed and the query substitution parameters.

Isolation Level

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

The queries and transactions were run with the isolation level set to "Level 3" (repeatable read).

Refresh Function Source Code

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

The refresh function is part of the implementation-specific driver code included in Appendix F.

Clause 3: Database System Properties

Atomicity

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

Completed Transaction

Perform the ACID Transaction (see Clause 3.1.5) for a randomly selected set of input data and verify that the appropriate rows have been changed in the ORDERS, LINEITEM, and HISTORY tables.

1. The total prices 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 was 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.

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, and were verified as not changed.

Consistency

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

Consistency Test

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

1. The consistency of the ORDER and LINEITEM tables was verified based on a sample of O_ORDERKEYs.
2. 100 ACID transactions were submitted from each of 2 execution streams.
3. The consistency of the ORDER and LINEITEM tables was verified a second time with the same O_ORDERKEYs.

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.

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. An 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.

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. An 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.

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. A second ACID transaction, T2, was started using the same O_KEY and L_KEY and a different randomly selected DELTA.
3. T2 waited.
4. T1 was allowed to commit and then T2 completed.
5. It was verified that T2.L_EXTENDPRICE was calculated correctly.

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. A second ACID transaction, T2, was started using the same O_KEY and L_KEY and a different randomly selected DELTA.
3. T2 waited.
4. T1 was allowed to rollback and then T2 completed.
5. It was verified that T2.L_EXTENDPRICE was calculated correctly.

Durability

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

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-R database tables or recovery log tables.

The database logs were stored on 4 disks using hardware RAID 1/0.

The tables for the database were also stored on drives using hardware RAID 1/0.

1. Two streams of ACID transactions were started.
2. After each stream had completed at least 100 transactions, one side of the mirrored set of logs was disabled.
3. The test continued to run with the loss of a log disk.
4. Then, a data disk was disabled. Since it was on a RAID 1/0, the test continued to run.
5. The database power plug was physically removed.
6. New drives were used to replace the disabled drives.
7. The RAID controllers rebuilt the logical volumes with the new drives.
8. The database ran through its recovery mode.
9. The counts in the success files and the HISTORY table count were compared. The counts matched.

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 durable medium failure, system crash test and the memory failure test were combined. See the previous section.

Memory Failure

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

The durable medium failure, system crash test and the memory failure test were combined. See the section above.

Clause 4: Scaling and Database Population

Initial 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.

Initial number of rows

Table	Occurrences
Orders	150,000,000
Lineitem	600,037,902
Customer	15,000,000
Part	20,000,000
Supplier	1,000,000
Partsupp	80,000,000
Nation	25
Region	5

Distribution of Tables and Logs Across Media

The distribution of tables and logs across all media must be explicitly described using a format similar to that shown in the following example for both the tested and priced systems.

***Comment**: Detailed diagrams for layout of database tables on disks can widely vary, and it is difficult to provide exact guidelines suitable for all implementations. The intent is to provide sufficient detail to allow independent reconstruction of the test database. The table below is an example of database layout descriptions and is not intended to describe any optimal layout for the TPC-R database.*

The database tables were distributed across 6 PowerVault 220S disk enclosures. Each enclosure contained two ESM modules. The arrays were configured as follows:

- Database files were stored on 6 RAID 1/0 volumes, each volume consisting of 14- 36GB 15k disks for a total 84 drives.
- The log files were stored on 1 RAID 1/0 volume, using 4 - 36GB 10k disks.
- Flat files were stored on the RAID 1/0 volumes that stored the database files.
- Database binaries were stored on 2 – 18GB 10k drives that were connected to the internal SCSI.
- The operating system was stored on 2 – 18GB 10k drives that were connected to the internal SCSI.

Distribution of Storage

Controller	Disk Device	Storage Content
1	sdc	1 RAID 1/0 for the database data files
2	sdd	1 RAID 1/0 for the database data files
3	sde	1 RAID 1/0 for the database data files
4	sdf	1 RAID 1/0 for the log data files
5	sdg	1 RAID 1/0 for the database data files
6	sdh	1 RAID 1/0 for the database data files
7	sdi	1 RAID 1/0 for the database data files
Onboard SCSI	Sda, sdb	O/S and DBMS files

Partitioning and Replication

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

***Comment**: The intent is to provide sufficient detail about partitioning and replication to allow independent reconstruction of the test database.*

All objects and their partitions were created in a single tablespace, and the RDBMS chose extents automatically from the datafiles that made up the tablespace. The tablespace consisted of 24 datafiles, laid out as 4 datafiles on each of the 6 RAID 1/0 volumes intended for database datafiles.

RAID Level

Implementations may use some form of RAID. The RAID level used must be disclosed for each device. If RAID is used in an implementation, the logical intent of its use must be disclosed. Three levels of usage are defined:
• Base tables only: In this case only the Base Tables (see Clause 1.2) are protected by any form of RAID;
• Base tables and auxiliary data structures: in addition to the protection of the base tables, implementations in this class must also employ RAID to protect all auxiliary data structures;
• Everything: implementations in this usage category must employ RAID to protect all database storage, including temporary or scratch space in addition to the base tables and auxiliary data structures.

RAID 1/0 was used on everything.

DBGEN Version and Modifications

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

DBGEN Version 1.3.0 was used for database population. Slight modifications were made to partition the output files and reorder the columns. These changes only affect how the data is organized in the flatfiles, but does not affect how the data is actually generated. The modified source code for DBGEN is disclosed in Appendix B.

Database Load time

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

Database load time was 14 hours 8 minutes 14 seconds.

Data Storage Ratio

The data storage ratio must be disclosed. It is computed by dividing the total data storage of the priced configuration (expressed in GB) by the size chosen for the test database as defined in Clause 4.1.3.1. The ratio must be reported to the nearest 1/100th, rounded up. For example, a system configured with 96 disks of 2.1 GB capacity for a 100GB test database has a data storage ratio of 2.02.

***Comment**: For the reporting of configured disk capacity, gigabyte (GB) is defined to be 2^30 bytes. Since disk manufacturers typically report disk size using base ten (i.e., GB = 10^9), it may be necessary to convert the advertised size from base ten to base two.*

Data Storage Ratio

Disk Type	Number of Disks	Space per Disk	Subtotal Disk Space
18GB	2	16.76GB	33.52GB
36GB	86	33.53GB	2950.64GB

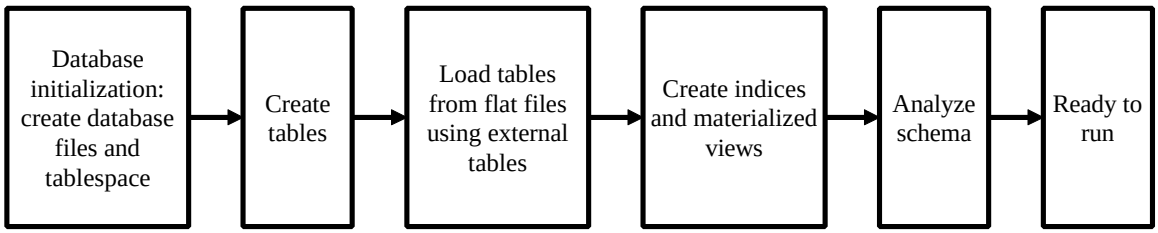
Total disk storage: 2984.16GB
Data storage ratio: 29.84

Database Load Mechanism Details and Illustration

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

DBGEN v1.3.0 was used to create flat files, which were then loaded into the database via external table mechanism. The flat files were split into multiple files per table (as documented in the load scripts in the appendix), except the REGION and NATION tables which each had only one file.

PERL scripts were used to perform the creation of the table spaces, tables, indexes and materialized views. It was also used to run the analyze scripts. The controller cache was set to 64 MB write cache and 64 MB read cache during the database load and the test runs.



Database Load Process

Differences between Qualification and Test Database

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

The qualification database used identical scripts to create and load the data with changes to adjust for the database scale factor.

Clause 5: Performance Metrics and Execution Rules

System Activity on SUT between Load and Performance Test

Any system activity on the SUT that takes place between the conclusion of the load test and the beginning of the performance test must be fully disclosed including listings of scripts or command logs.

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

The SUT was rebooted and the database was restarted.

All scripts and queries used are included in Appendix F.

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. RF1 update transaction.
2. Stream 00 execution.
3. RF2 update transaction.

Timing Intervals

The timing intervals (see Clause 5.3.7) 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 disclosed in the Numerical Quantities Summary at the beginning of this document.

Number of Streams for The Throughput Test

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

5 execution streams were used for the throughput test.

Start/Finish Time of 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 disclosed in the Numerical Quantities Summary earlier in this document.

Total Elapsed Time

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

The total elapsed time of the throughput test was 25573 seconds.

Start/Finish Time for Refresh Function

Start and finish time for each refresh 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 disclosed in the Numerical Quantities Summary earlier in this document.

Timing Intervals for Each Query and Each Update

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

The timing intervals for each query and each update function are contained in the Numerical Quantities Summary disclosed earlier in this document.

Computed Performance Metrics

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

The performance metrics, and the numbers, on which they are based, are contained in the Numerical Quantities section of the Executive Summary.

Performance Metrics

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

Run 1 and 2 Results from Benchmark Executions

Run	QphR@100GB	TPC-R Power@100GB	TPC-R Throughput@100GB
1	4862.9	14277.7	1656.3
2	4452.5	12802.6	1548.5

System Activity between Run1 and Run2

Any activity on the SUT that takes place between the conclusion of Run1 and the beginning of Run2 must be fully disclosed including listings of scripts or command logs along with any system reboots or database restarts.

The system was rebooted and the database was restarted between the two runs.

Clause 6: SUT and Driver Implementation

Driver

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

A single script performs all stream executions. QGEN is used to produce query text.
For each power-test run:

The SQL for RF1 is submitted to the database
Then the queries as generated by QGEN are submitted in the order defined by Clause 5.3.5.4
The SQL for RF2 is submitted to the database**.**

There is one script that creates and loads the database. The other script runs the power and throughput test. These scripts are disclosed in Appendix F.

Implementation Specific Layer (ISL)

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

The source code for the "qexec" utility can be found in Appendix F.

Profile-Directed Optimization

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

Profile-directed optimization subject to the requirements of *5.2.9* and *5.2.10* was not used.

Clause 7: Pricing

Hardware and Software Used in the Priced System

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

A detailed list of the hardware and software used in the priced system is included in the Executive Summary at the beginning of this document.

Total Three year Price

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

A detailed price sheet of all the hardware and software used in this configuration, including the 3-year maintenance cost, and total price, is included in the executive summary at the beginning of this document.

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 availability date reported on the executive summary must be the date by which all components are committed to being available. The full disclosure report must report availability dates individually for at least each of the categories for which a pricing subtotal must be provided.

All components are available as of the date reported in the executive summary of this document.

Clause 8: Auditor-Related Items

Auditor's Report

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

This implementation of the TPC Benchmark R was audited by Lorna Livingtree of Performance Metrics Incorporated. Further information can be obtained from:

Performance Metrics Inc.
137 Yankton St. Suite 101
Folsom CA 95630
Phone: (916) 985-1131
Fax: (916) 985-1185
Email: Lorna@PerfMetrics.com

The auditor's letter of attestation is included in this full disclosure report as Appendix I.

Appendix A: Database Parameter Settings

System Configuration Settings

Rc.local

```
#!/bin/sh
#
# This script will be executed *after* all the
other init scripts.
# You can put your own initialization stuff in
here if you don't
# want to do the full Sys V style init stuff.

touch /var/lock/subsys/local



#100G
/oracle/scripts-raw/rawmap.sh

# 32K
echo 32768 > /proc/sys/fs/file-max
# 512M
#echo 536870912 > /proc/sys/kernel/shmmax
# 3GB
echo 3221225472 > /proc/sys/kernel/shmmax

# set aio-max-size to >=1M, here its 2M, size of
memory allocation
echo 2097152 > /proc/sys/fs/aio-max-size

/usr/bin/setterm -blank 0
```

cpuinfo.out

```
processor       : 0
vendor_id       : GenuineIntel
cpu family      : 15
model           : 1
model name      : Intel(R) Genuine CPU 1.60GHz
stepping        : 1
cpu MHz                 : 1594.890
cache size      : 256 KB
fdiv_bug        : no
hlt_bug                 : no
f00f_bug        : no
coma_bug        : no
fpu             : yes
fpu_exception : yes
cpuid level     : 2
wp              : yes
flags           : fpu vme de pse tsc msr pae mce
cx8 apic sep mtrr pge mca cmov pat pse36
clflush dts acpi mmx fxsr sse sse2 ss ht tm
bogomips        : 3185.04

processor       : 1
vendor_id       : GenuineIntel
cpu family      : 15
model           : 1
model name      : Intel(R) Genuine CPU 1.60GHz
stepping        : 1
cpu MHz                 : 1594.890
cache size      : 256 KB
fdiv_bug        : no
hlt_bug                 : no
f00f_bug        : no
coma_bug        : no
fpu             : yes
fpu_exception : yes
cpuid level     : 2
wp              : yes
flags           : fpu vme de pse tsc msr pae mce
cx8 apic sep mtrr pge mca cmov pat pse36
clflush dts acpi mmx fxsr sse sse2 ss ht tm
bogomips        : 3185.04

processor       : 2
vendor_id       : GenuineIntel
cpu family      : 15
model           : 1
model name      : Intel(R) Genuine CPU 1.60GHz
stepping        : 1
cpu MHz                 : 1594.890
cache size      : 256 KB
fdiv_bug        : no
hlt_bug                 : no
f00f_bug        : no
coma_bug        : no
fpu             : yes
fpu_exception : yes
cpuid level     : 2
wp              : yes
```

flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm
bogomips : 3185.04

processor : 3
vendor_id : GenuineIntel
cpu family : 15
model : 1
model name : Intel(R) Genuine CPU 1.60GHz
stepping : 1
cpu MHz : 1594.890
cache size : 256 KB
fdiv_bug : no
hlt_bug : no
f00f_bug : no
coma_bug : no
fpu : yes
fpu_exception : yes
cpuid level : 2
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm
bogomips : 3185.04

processor : 4
vendor_id : GenuineIntel
cpu family : 15
model : 1
model name : Intel(R) Genuine CPU 1.60GHz
stepping : 1
cpu MHz : 1594.890
cache size : 256 KB
fdiv_bug : no
hlt_bug : no
f00f_bug : no
coma_bug : no
fpu : yes
fpu_exception : yes
cpuid level : 2
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm
bogomips : 3185.04

processor : 5
vendor_id : GenuineIntel
cpu family : 15
model : 1
model name : Intel(R) Genuine CPU 1.60GHz
stepping : 1
cpu MHz : 1594.890
cache size : 256 KB
fdiv_bug : no
hlt_bug : no
f00f_bug : no
coma_bug : no
fpu : yes
fpu_exception : yes
cpuid level : 2
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm
bogomips : 3185.04

processor : 6
vendor_id : GenuineIntel
cpu family : 15
model : 1
model name : Intel(R) Genuine CPU 1.60GHz
stepping : 1
cpu MHz : 1594.890
cache size : 256 KB
fdiv_bug : no
hlt_bug : no
f00f_bug : no
coma_bug : no
fpu : yes
fpu_exception : yes
cpuid level : 2
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm
bogomips : 3185.04

processor : 7
vendor_id : GenuineIntel
cpu family : 15
model : 1
model name : Intel(R) Genuine CPU 1.60GHz
stepping : 1
cpu MHz : 1594.890

cache size : 256 KB
fdiv_bug : no
hlt_bug : no
f00f_bug : no
coma_bug : no
fpu : yes
fpu_exception : yes
cpuid level : 2
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm
bogomips : 3185.04

meminfo.out

total: used: free: shared: buffers: cached:
Mem: 3956015104 2756661248 1199353856 257261568 367927296 1769472000
Swap: 2146787328 0 2146787328
MemTotal: 3863296 kB
MemFree: 1171244 kB
MemShared: 251232 kB
Buffers: 359304 kB
Cached: 1728000 kB
SwapCached: 0 kB
Active: 122284 kB
Inact_dirty: 2144100 kB
Inact_clean: 72152 kB
Inact_target: 983024 kB
HighTotal: 3014592 kB
HighFree: 1054184 kB
LowTotal: 848704 kB
LowFree: 117060 kB
SwapTotal: 2096472 kB
SwapFree: 2096472 kB
BigPagesFree: 0 kB

rawmap.sh

raw /dev/raw/raw4 /dev/sdc1
chown oracle:dba /dev/raw/raw4
chmod 660 /dev/raw/raw4
raw /dev/raw/raw10 /dev/sdc2
chown oracle:dba /dev/raw/raw10
chmod 660 /dev/raw/raw10
raw /dev/raw/raw16 /dev/sdc3
chown oracle:dba /dev/raw/raw16
chmod 660 /dev/raw/raw16
raw /dev/raw/raw22 /dev/sdc5
chown oracle:dba /dev/raw/raw22
chmod 660 /dev/raw/raw22
raw /dev/raw/raw28 /dev/sdc6
chown oracle:dba /dev/raw/raw28
chmod 660 /dev/raw/raw28
raw /dev/raw/raw34 /dev/sdc7
chown oracle:dba /dev/raw/raw34
chmod 660 /dev/raw/raw34
raw /dev/raw/raw40 /dev/sdc8
chown oracle:dba /dev/raw/raw40
chmod 660 /dev/raw/raw40
raw /dev/raw/raw1 /dev/sdc9
chown oracle:dba /dev/raw/raw1
chmod 660 /dev/raw/raw1
raw /dev/raw/raw100 /dev/sdc10
chown oracle:dba /dev/raw/raw100
chmod 660 /dev/raw/raw100
raw /dev/raw/raw55 /dev/sdc11
chown oracle:dba /dev/raw/raw55
chmod 660 /dev/raw/raw55
raw /dev/raw/raw61 /dev/sdc12
chown oracle:dba /dev/raw/raw61
chmod 660 /dev/raw/raw61
raw /dev/raw/raw5 /dev/sdd1
chown oracle:dba /dev/raw/raw5
chmod 660 /dev/raw/raw5
raw /dev/raw/raw11 /dev/sdd2
chown oracle:dba /dev/raw/raw11
chmod 660 /dev/raw/raw11
raw /dev/raw/raw17 /dev/sdd3
chown oracle:dba /dev/raw/raw17
chmod 660 /dev/raw/raw17
raw /dev/raw/raw23 /dev/sdd5
chown oracle:dba /dev/raw/raw23
chmod 660 /dev/raw/raw23
raw /dev/raw/raw29 /dev/sdd6
chown oracle:dba /dev/raw/raw29
chmod 660 /dev/raw/raw29
raw /dev/raw/raw35 /dev/sdd7
chown oracle:dba /dev/raw/raw35
chmod 660 /dev/raw/raw35
raw /dev/raw/raw41 /dev/sdd8
chown oracle:dba /dev/raw/raw41
chmod 660 /dev/raw/raw41
raw /dev/raw/raw2 /dev/sdd9

```
chown oracle:dba /dev/raw/raw2
chmod 660 /dev/raw/raw2
raw /dev/raw/raw101 /dev/sdd10
chown oracle:dba /dev/raw/raw101
chmod 660 /dev/raw/raw101
raw /dev/raw/raw56 /dev/sdd11
chown oracle:dba /dev/raw/raw56
chmod 660 /dev/raw/raw56
raw /dev/raw/raw62 /dev/sdd12
chown oracle:dba /dev/raw/raw62
chmod 660 /dev/raw/raw62
raw /dev/raw/raw6 /dev/sde1
chown oracle:dba /dev/raw/raw6
chmod 660 /dev/raw/raw6
raw /dev/raw/raw12 /dev/sde2
chown oracle:dba /dev/raw/raw12
chmod 660 /dev/raw/raw12
raw /dev/raw/raw18 /dev/sde3
chown oracle:dba /dev/raw/raw18
chmod 660 /dev/raw/raw18
raw /dev/raw/raw24 /dev/sde5
chown oracle:dba /dev/raw/raw24
chmod 660 /dev/raw/raw24
raw /dev/raw/raw30 /dev/sde6
chown oracle:dba /dev/raw/raw30
chmod 660 /dev/raw/raw30
raw /dev/raw/raw36 /dev/sde7
chown oracle:dba /dev/raw/raw36
chmod 660 /dev/raw/raw36
raw /dev/raw/raw42 /dev/sde8
chown oracle:dba /dev/raw/raw42
chmod 660 /dev/raw/raw42
raw /dev/raw/raw3 /dev/sde9
chown oracle:dba /dev/raw/raw3
chmod 660 /dev/raw/raw3
raw /dev/raw/raw102 /dev/sde10
chown oracle:dba /dev/raw/raw102
chmod 660 /dev/raw/raw102
raw /dev/raw/raw57 /dev/sde11
chown oracle:dba /dev/raw/raw57
chmod 660 /dev/raw/raw57
raw /dev/raw/raw46 /dev/sdf1
chown oracle:dba /dev/raw/raw46
chmod 660 /dev/raw/raw46
raw /dev/raw/raw47 /dev/sdf2
chown oracle:dba /dev/raw/raw47
chmod 660 /dev/raw/raw47
raw /dev/raw/raw50 /dev/sdf3
chown oracle:dba /dev/raw/raw50
chmod 660 /dev/raw/raw50
raw /dev/raw/raw51 /dev/sdf5
chown oracle:dba /dev/raw/raw51
chmod 660 /dev/raw/raw51
raw /dev/raw/raw7 /dev/sdg1
chown oracle:dba /dev/raw/raw7
chmod 660 /dev/raw/raw7
raw /dev/raw/raw13 /dev/sdg2
chown oracle:dba /dev/raw/raw13
chmod 660 /dev/raw/raw13
raw /dev/raw/raw19 /dev/sdg3
chown oracle:dba /dev/raw/raw19
chmod 660 /dev/raw/raw19
raw /dev/raw/raw25 /dev/sdg5
chown oracle:dba /dev/raw/raw25
chmod 660 /dev/raw/raw25
raw /dev/raw/raw31 /dev/sdg6
chown oracle:dba /dev/raw/raw31
chmod 660 /dev/raw/raw31
raw /dev/raw/raw37 /dev/sdg7
chown oracle:dba /dev/raw/raw37
chmod 660 /dev/raw/raw37
raw /dev/raw/raw43 /dev/sdg8
chown oracle:dba /dev/raw/raw43
chmod 660 /dev/raw/raw43
raw /dev/raw/raw52 /dev/sdg9
chown oracle:dba /dev/raw/raw52
chmod 660 /dev/raw/raw52
raw /dev/raw/raw103 /dev/sdg10
chown oracle:dba /dev/raw/raw103
chmod 660 /dev/raw/raw103
raw /dev/raw/raw58 /dev/sdg11
chown oracle:dba /dev/raw/raw58
chmod 660 /dev/raw/raw58
raw /dev/raw/raw8 /dev/sdh1
chown oracle:dba /dev/raw/raw8
chmod 660 /dev/raw/raw8
raw /dev/raw/raw14 /dev/sdh2
chown oracle:dba /dev/raw/raw14
chmod 660 /dev/raw/raw14
raw /dev/raw/raw20 /dev/sdh3
chown oracle:dba /dev/raw/raw20
chmod 660 /dev/raw/raw20
raw /dev/raw/raw26 /dev/sdh5
chown oracle:dba /dev/raw/raw26
chmod 660 /dev/raw/raw26
raw /dev/raw/raw32 /dev/sdh6
```

```
chown oracle:dba /dev/raw/raw32
chmod 660 /dev/raw/raw32
raw /dev/raw/raw38 /dev/sdh7
chown oracle:dba /dev/raw/raw38
chmod 660 /dev/raw/raw38
raw /dev/raw/raw44 /dev/sdh8
chown oracle:dba /dev/raw/raw44
chmod 660 /dev/raw/raw44
raw /dev/raw/raw53 /dev/sdh9
chown oracle:dba /dev/raw/raw53
chmod 660 /dev/raw/raw53
raw /dev/raw/raw104 /dev/sdh10
chown oracle:dba /dev/raw/raw104
chmod 660 /dev/raw/raw104
raw /dev/raw/raw59 /dev/sdh11
chown oracle:dba /dev/raw/raw59
chmod 660 /dev/raw/raw59
raw /dev/raw/raw9 /dev/sdi1
chown oracle:dba /dev/raw/raw9
chmod 660 /dev/raw/raw9
raw /dev/raw/raw15 /dev/sdi2
chown oracle:dba /dev/raw/raw15
chmod 660 /dev/raw/raw15
raw /dev/raw/raw21 /dev/sdi3
chown oracle:dba /dev/raw/raw21
chmod 660 /dev/raw/raw21
raw /dev/raw/raw27 /dev/sdi5
chown oracle:dba /dev/raw/raw27
chmod 660 /dev/raw/raw27
raw /dev/raw/raw33 /dev/sdi6
chown oracle:dba /dev/raw/raw33
chmod 660 /dev/raw/raw33
raw /dev/raw/raw39 /dev/sdi7
chown oracle:dba /dev/raw/raw39
chmod 660 /dev/raw/raw39
raw /dev/raw/raw45 /dev/sdi8
chown oracle:dba /dev/raw/raw45
chmod 660 /dev/raw/raw45
raw /dev/raw/raw54 /dev/sdi9
chown oracle:dba /dev/raw/raw54
chmod 660 /dev/raw/raw54
raw /dev/raw/raw105 /dev/sdi10
chown oracle:dba /dev/raw/raw105
chmod 660 /dev/raw/raw105
raw /dev/raw/raw60 /dev/sdi11
chown oracle:dba /dev/raw/raw60
chmod 660 /dev/raw/raw60
```

Oracle Parameter Settings

inittpcr100g.ora

```
audit_trail                    = FALSE
compatible                     = 9.0.1
control_files                  =
($ORACLE_HOME/dbs/tpcr_disks/ctrlfile_1,
$ORACLE_HOME/dbs/tpcr_disks/ctrlfile_2)
db_block_checksum              = false
db_block_size                  = 8192
db_cache_size                  = 1200m
db_file_multiblock_read_count  = 128
db_files                       = 120
db_name                        = tpcr100g
global_names                   = FALSE
java_pool_size                 = 0
large_pool_size                = 90000000
log_buffer                     = 33554432
log_checkpoints_to_alert       = true
max_dump_file_size             = 5000
nls_date_format                = YYYY-MM-DD
open_cursors                   = 1024
optimizer_mode                 = CHOOSE
parallel_automatic_tuning      = true
parallel_execution_message_size = 8192
parallel_max_servers           = 256
parallel_min_servers           = 128
pga_aggregate_target           = 2000M
processes                      = 256
query_rewrite_enabled          = true
recovery_parallelism           = 4
sessions                       = 300
shared_pool_size               = 90000000
statistics_level               = basic
transactions                   = 512
undo_management                = AUTO
undo_retention                 = 86400
undo_tablespace                = ts_undo
```

Appendix B: Programs and Scripts

NOTE: Some of the scripts are very long, and, in order to save space, are printed in very small font. The electronic version of the FDR has the complete listings. But all of the scripts may not be readable in the paper version.

Load100g_data.dat

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

%b-preproc

*sql
\echo "{}" > script*getenv(BUMPX_CTR).sql
\sqlplus /NOLOG <<!
\set echo on;
\set termout on;
\connect / as sysdba;
\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 {}
\/oracle/kit/dbgen/dbgen {}

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

*sh
\{}

%e-preproc
%b-dbcre
*bgon=1
################################################
############################
# Database Creation Phase
*sql
{
shutdown abort;
}
*wait
# creating database and initial rollback segment
*time{Database create start}
*sql
{
startup pfile=/oracle/oracle9i/dbs/inittpcr100g.ora
nomount;
create database
  controlfile reuse
  logfile
   '/oracle/oracle9i/dbs/tpcr_disks/log1' size 32767m reuse,
   '/oracle/oracle9i/dbs/tpcr_disks/log2' size 32767m reuse
  datafile '/oracle/oracle9i/dbs/tpcr_disks/system' size
500m reuse
  undo tablespace ts_undo
            datafile
        '/oracle/oracle9i/dbs/tpcr_disks/ts_undo_1' size
14336m reuse,
             '/oracle/oracle9i/dbs/tpcr_disks/ts_undo_2' size
14336m reuse,
             '/oracle/oracle9i/dbs/tpcr_disks/ts_undo_3' size
14336m reuse,
             '/oracle/oracle9i/dbs/tpcr_disks/ts_undo_4' size
14336m reuse,
             '/oracle/oracle9i/dbs/tpcr_disks/ts_undo_5' size
14336m reuse,
             '/oracle/oracle9i/dbs/tpcr_disks/ts_undo_6' size
14336m reuse
  maxdatafiles 1024
;
}
*time{Database create end}
*wait
# building data dictionary
*sql
{
set termout off
set echo off
spool /tmp/cat.out
```

```
@/oracle/oracle9i/rdbms/admin/catalog.sql;
@/oracle/oracle9i/rdbms/admin/catparr.sql;
@/oracle/oracle9i/rdbms/admin/catproc.sql;
@/oracle/oracle9i/rdbms/admin/utlxplan;
connect system/manager;
@/oracle/oracle9i/sqlplus/admin/pupbld.sql;
spool off
}
*wait
*bgoff
%e-dbcre
%b-sctso
*bgon=1
################################################
################################
# Schema Creation Phase - datafiles only (no tables or
users)
# creating data tablespaces, datafiles
#
*time{Schema create start - adding undo, data and temp
files only}
#
# creating tpcd's ts_data tablespace
*sql
{
rem drop tablespace ts_data;
create tablespace ts_data
datafile '/oracle/oracle9i/dbs/tpcr_disks/ts_data_1' size
32767m reuse
extent management dictionary
nologging
;
}
*wait
# creating tpcd's ts_temp tablespace
*sql
{
rem drop tablespace ts_temp;
create temporary tablespace ts_temp
tempfile '/oracle/oracle9i/dbs/tpcr_disks/ts_temp_1' size
32767m reuse
extent management local
uniform size 10m
;
}
*wait
# adding extra files to tpcd's ts_data tablespace
*sql
{
alter tablespace ts_data add
datafile '/oracle/oracle9i/dbs/tpcr_disks/ts_data_2' size
32767m reuse;
}
*sql
{
alter tablespace ts_data add
datafile '/oracle/oracle9i/dbs/tpcr_disks/ts_data_3' size
32767m reuse;
}
*sql
{
alter tablespace ts_data add
datafile '/oracle/oracle9i/dbs/tpcr_disks/ts_data_4' size
32767m reuse;
}
*wait
*sql
{
alter tablespace ts_data add
datafile '/oracle/oracle9i/dbs/tpcr_disks/ts_data_5' size
32767m reuse;
}
*sql
{
alter tablespace ts_data add
datafile '/oracle/oracle9i/dbs/tpcr_disks/ts_data_6' size
32767m reuse;
}
*sql
{
alter tablespace ts_data add
datafile '/oracle/oracle9i/dbs/tpcr_disks/ts_data_7' size
32767m reuse;
}
*sql
{
alter tablespace ts_data add
datafile '/oracle/oracle9i/dbs/tpcr_disks/ts_data_8' size
32767m reuse;
}
*wait
*sql
{
alter tablespace ts_data add
datafile '/oracle/oracle9i/dbs/tpcr_disks/ts_data_9' size
32767m reuse;
}
*sql
{
alter tablespace ts_data add
datafile '/oracle/oracle9i/dbs/tpcr_disks/ts_data_10' size
32767m reuse;
}
*sql
{
alter tablespace ts_data add
datafile '/oracle/oracle9i/dbs/tpcr_disks/ts_data_11' size
32767m reuse;
}
*sql
{
alter tablespace ts_data add
datafile '/oracle/oracle9i/dbs/tpcr_disks/ts_data_12' size
32767m reuse;
}
*wait
```

```
*sql
{
alter tablespace ts_data add
datafile '/oracle/oracle9i/dbs/tpcr_disks/ts_data_13' size
32767m reuse;
}
*sql
{
alter tablespace ts_data add
datafile '/oracle/oracle9i/dbs/tpcr_disks/ts_data_14' size
32767m reuse;
}
*sql
{
alter tablespace ts_data add
datafile '/oracle/oracle9i/dbs/tpcr_disks/ts_data_15' size
32767m reuse;
}
*sql
{
alter tablespace ts_data add
datafile '/oracle/oracle9i/dbs/tpcr_disks/ts_data_16' size
32767m reuse;
}
*wait
*sql
{
alter tablespace ts_data add
datafile '/oracle/oracle9i/dbs/tpcr_disks/ts_data_17' size
32767m reuse;
}
*sql
{
alter tablespace ts_data add
datafile '/oracle/oracle9i/dbs/tpcr_disks/ts_data_18' size
32767m reuse;
}
*sql
{
alter tablespace ts_data add
datafile '/oracle/oracle9i/dbs/tpcr_disks/ts_data_19' size
32767m reuse;
}
*sql
{
alter tablespace ts_data add
datafile '/oracle/oracle9i/dbs/tpcr_disks/ts_data_20' size
32767m reuse;
}
*wait
*sql
{
alter tablespace ts_data add
datafile '/oracle/oracle9i/dbs/tpcr_disks/ts_data_21' size
32767m reuse;
}
*sql
{
alter tablespace ts_data add
datafile '/oracle/oracle9i/dbs/tpcr_disks/ts_data_22' size
32767m reuse;
}
*sql
{
alter tablespace ts_data add
datafile '/oracle/oracle9i/dbs/tpcr_disks/ts_data_23' size
32767m reuse;
}
*sql
{
alter tablespace ts_data add
datafile '/oracle/oracle9i/dbs/tpcr_disks/ts_data_24' size
32767m reuse;
}
*wait
# adding extra files to tpcd's ts_temp tablespace
*sql
{
alter tablespace ts_temp
  add tempfile '/oracle/oracle9i/dbs/tpcr_disks/ts_temp_2'
size 32767m reuse;
}
*sql
{
alter tablespace ts_temp
  add tempfile '/oracle/oracle9i/dbs/tpcr_disks/ts_temp_3'
size 32767m reuse;
}
*sql
{
alter tablespace ts_temp
  add tempfile '/oracle/oracle9i/dbs/tpcr_disks/ts_temp_4'
size 32767m reuse;
}
*sql
{
alter tablespace ts_temp
  add tempfile '/oracle/oracle9i/dbs/tpcr_disks/ts_temp_5'
size 32767m reuse;
}
*wait
*sql
{
alter tablespace ts_temp
  add tempfile '/oracle/oracle9i/dbs/tpcr_disks/ts_temp_6'
size 32767m reuse;
}
*sql
{
alter tablespace ts_temp
  add tempfile '/oracle/oracle9i/dbs/tpcr_disks/ts_temp_7'
size 32767m reuse;
}
*sql
{
alter tablespace ts_temp
```

```
  add tempfile '/oracle/oracle9i/dbs/tpcr_disks/ts_temp_8'
size 32767m reuse;
}
*sql
{
alter tablespace ts_temp
  add tempfile '/oracle/oracle9i/dbs/tpcr_disks/ts_temp_9'
size 32767m reuse;
}
*wait
*sql
{
alter tablespace ts_temp
  add tempfile '/oracle/oracle9i/dbs/tpcr_disks/ts_temp_10'
size 32767m reuse;
}
*sql
{
alter tablespace ts_temp
  add tempfile '/oracle/oracle9i/dbs/tpcr_disks/ts_temp_11'
size 32767m reuse;
}
*sql
{
alter tablespace ts_temp
  add tempfile '/oracle/oracle9i/dbs/tpcr_disks/ts_temp_12'
size 32767m reuse;
}
*wait
*time{Schema create end - datafiles only}
*bgoff
%e-sctso
%b-dapop
*bgon=1
################################################
################################
# Schema Creation Phase - User and Tables ONLY (no
datafiles)
# creating tpcd user
*time{Schema create start - User & Tables, i.e., Load
Start}
*sql
{
drop user tpcd cascade;
grant DBA,query rewrite,global query rewrite
  to tpcd identified by tpcd;
}
# altering tpcd's temp tablespace
*sql
{
alter user tpcd temporary tablespace ts_temp;
alter user tpcd default tablespace ts_data;
}
*sql
{
connect tpcd/tpcd
@/oracle/oracle9i/rdbms/admin/utlxplan.sql;
}
*wait
# Database Population Phase
*sql
{
connect tpcd/tpcd
set echo on
set timing on

drop directory data_dir;
create directory data_dir as '/flat_files';

rem drop table l_et;
create table l_et(
    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          char(10) ,
    l_linenumber        number ,
    l_shipinstruct      char(25) ,
    l_comment           varchar(44)
)
organization external (
type ORACLE_LOADER
default directory data_dir
access parameters
(
                records delimited by newline
                badfile 'l_et.bad'
                logfile 'l_et.log'
                fields terminated by '|'
                missing field values are null
        )
        location (
        'lineitem.tbl.1',
        'lineitem.tbl.2',
        'lineitem.tbl.3',
        'lineitem.tbl.4',
        'lineitem.tbl.5',
        'lineitem.tbl.6',
        'lineitem.tbl.7',
        'lineitem.tbl.8',
        'lineitem.tbl.9',
        'lineitem.tbl.10',
        'lineitem.tbl.11',
        'lineitem.tbl.12',
        'lineitem.tbl.13',
        'lineitem.tbl.14',
        'lineitem.tbl.15',
        'lineitem.tbl.16'
```

```
        ))
reject limit unlimited;

rem drop table o_et;
create table o_et(
   o_orderdate         date ,
   o_orderkey          number ,
   o_custkey           number ,
   o_orderpriority      char(15) ,
   o_shippriority      number ,
   o_clerk             char(15) ,
   o_orderstatus       char(1) ,
   o_totalprice        number ,
   o_comment           varchar(79)
)
organization external (
type ORACLE_LOADER
default directory data_dir
access parameters
(
                records delimited by newline
                badfile 'o_et.bad'
                logfile 'o_et.log'
                fields terminated by '|'
                missing field values are null
        )
        location (
        'orders.tbl.1',
        'orders.tbl.2',
        'orders.tbl.3',
        'orders.tbl.4',
        'orders.tbl.5',
        'orders.tbl.6',
        'orders.tbl.7',
        'orders.tbl.8',
        'orders.tbl.9',
        'orders.tbl.10',
        'orders.tbl.11',
        'orders.tbl.12',
        'orders.tbl.13',
        'orders.tbl.14',
        'orders.tbl.15',
        'orders.tbl.16'
      ))
reject limit unlimited;

rem drop table ps_et;
create table ps_et(
   ps_partkey          number ,
   ps_suppkey          number ,
   ps_supplycost       number ,
   ps_availqty         number ,
   ps_comment          varchar(199)
)
organization external (
type ORACLE_LOADER
default directory data_dir
access parameters
(
                records delimited by newline
                badfile 'ps_et.bad'
                logfile 'ps_et.log'
                fields terminated by '|'
                missing field values are null
        )
        location (
        'partsupp.tbl.1',
        'partsupp.tbl.2',
        'partsupp.tbl.3',
        'partsupp.tbl.4',
        'partsupp.tbl.5',
        'partsupp.tbl.6',
        'partsupp.tbl.7',
        'partsupp.tbl.8'
        ))
reject limit unlimited;

rem drop table p_et;
create table p_et(
   p_partkey           number ,
   p_type              varchar(25) ,
   p_size              number ,
   p_brand             char(10) ,
   p_name              varchar(55) ,
   p_container         char(10) ,
   p_mfgr              char(25) ,
   p_retailprice       number ,
   p_comment           varchar(23)
)
organization external (
type ORACLE_LOADER
default directory data_dir
access parameters
(
                records delimited by newline
                badfile 'p_et.bad'
                logfile 'p_et.log'
                fields terminated by '|'
                missing field values are null
        )
        location (
        'part.tbl.1',
        'part.tbl.2',
        'part.tbl.3',
        'part.tbl.4',
        'part.tbl.5',
        'part.tbl.6',
        'part.tbl.7',
        'part.tbl.8'
        ))
reject limit unlimited;

rem drop table c_et;
create table c_et(
   c_custkey           number ,
   c_mktsegment        char(10) ,
```

```
   c_nationkey          number ,
   c_name               varchar(25) ,
   c_address            varchar(40) ,
   c_phone              char(15) ,
   c_acctbal            number ,
   c_comment            varchar(117)
)
organization external (
type ORACLE_LOADER
default directory data_dir
access parameters
(
                        records delimited by newline
                        badfile 'c_et.bad'
                        logfile 'c_et.log'
                        fields terminated by '|'
                        missing field values are null
            )
            location (
            'customer.tbl.1',
            'customer.tbl.2',
            'customer.tbl.3',
            'customer.tbl.4',
            'customer.tbl.5',
            'customer.tbl.6',
            'customer.tbl.7',
            'customer.tbl.8'
            ))
reject limit unlimited;

rem drop table s_et;
create table s_et(
   s_suppkey            number ,
   s_nationkey          number ,
   s_comment            varchar(101) ,
   s_name               char(25) ,
   s_address            varchar(40) ,
   s_phone              char(15) ,
   s_acctbal            number
)
organization external (
type ORACLE_LOADER
default directory data_dir
access parameters
(
                        records delimited by newline
                        badfile 's_et.bad'
                        logfile 's_et.log'
                        fields terminated by '|'
                        missing field values are null
            )
            location (
            'supplier.tbl'))
reject limit unlimited;

rem drop table n_et;
create table n_et(
   n_nationkey          number ,
   n_name               char(25) ,
   n_regionkey          number ,
   n_comment            varchar(152)
)
organization external (
type ORACLE_LOADER
default directory data_dir
access parameters
(
                        records delimited by newline
                        badfile 'n_et.bad'
                        logfile 'n_et.log'
                        fields terminated by '|'
                        missing field values are null
            )
            location (
            'nation.tbl'))
reject limit unlimited;

rem drop table r_et;
create table r_et(
   r_regionkey          number ,
   r_name               char(25) ,
   r_comment            varchar(152)
)
organization external (
type ORACLE_LOADER
default directory data_dir
access parameters
(
                        records delimited by newline
                        badfile 'r_et.bad'
                        logfile 'r_et.log'
                        fields terminated by '|'
                        missing field values are null
            )
            location (
            'region.tbl'))
reject limit unlimited;

alter table l_et parallel 8;
alter table o_et parallel 8;
alter table ps_et parallel 8;
alter table p_et parallel 8;
alter table c_et parallel 8;
alter table s_et parallel 8;
}
*wait
*sql
{
connect tpcd/tpcd
set timing on
set echo on

rem drop table lineitem;
create table lineitem(
   l_shipdate           ,
   l_orderkey           NOT NULL,
```

```
    l_discount          ,
    l_extendedprice     ,
    l_suppkey           NOT NULL,
    l_quantity          ,
    l_returnflag        ,
    l_partkey           NOT NULL,
    l_linestatus        ,
    l_tax               NOT NULL,
    l_commitdate        ,
    l_receiptdate       ,
    l_shipmode          ,
    l_linenumber        NOT NULL,
    l_shipinstruct      ,
    l_comment
)
pctfree 1
storage(freelists 16 initial 22640k next 22640k
pctincrease 0 minextents 6  maxextents unlimited)
parallel 8
nologging
initrans 10
partition by range (l_shipdate)
subpartition by hash (l_partkey)
subpartitions 8
(
partition item1 values less than (to_date('1992-01-
01','YYYY-MM-DD'))
,
partition item2 values less than (to_date('1992-02-
01','YYYY-MM-DD'))
,
partition item3 values less than (to_date('1992-03-
01','YYYY-MM-DD'))
,
partition item4 values less than (to_date('1992-04-
01','YYYY-MM-DD'))
,
partition item5 values less than (to_date('1992-05-
01','YYYY-MM-DD'))
,
partition item6 values less than (to_date('1992-06-
01','YYYY-MM-DD'))
,
partition item7 values less than (to_date('1992-07-
01','YYYY-MM-DD'))
,
partition item8 values less than (to_date('1992-08-
01','YYYY-MM-DD'))
,
partition item9 values less than (to_date('1992-09-
01','YYYY-MM-DD'))
,
partition item10 values less than (to_date('1992-10-
01','YYYY-MM-DD'))
,
partition item11 values less than (to_date('1992-11-
01','YYYY-MM-DD'))
,
partition item12 values less than (to_date('1992-12-
01','YYYY-MM-DD'))
,
partition item13 values less than (to_date('1993-01-
01','YYYY-MM-DD'))
,
partition item14 values less than (to_date('1993-02-
01','YYYY-MM-DD'))
,
partition item15 values less than (to_date('1993-03-
01','YYYY-MM-DD'))
,
partition item16 values less than (to_date('1993-04-
01','YYYY-MM-DD'))
,
partition item17 values less than (to_date('1993-05-
01','YYYY-MM-DD'))
,
partition item18 values less than (to_date('1993-06-
01','YYYY-MM-DD'))
,
partition item19 values less than (to_date('1993-07-
01','YYYY-MM-DD'))
,
partition item20 values less than (to_date('1993-08-
01','YYYY-MM-DD'))
,
partition item21 values less than (to_date('1993-09-
01','YYYY-MM-DD'))
,
partition item22 values less than (to_date('1993-10-
01','YYYY-MM-DD'))
,
partition item23 values less than (to_date('1993-11-
01','YYYY-MM-DD'))
,
partition item24 values less than (to_date('1993-12-
01','YYYY-MM-DD'))
,
partition item25 values less than (to_date('1994-01-
01','YYYY-MM-DD'))
,
partition item26 values less than (to_date('1994-02-
01','YYYY-MM-DD'))
,
partition item27 values less than (to_date('1994-03-
01','YYYY-MM-DD'))
,
partition item28 values less than (to_date('1994-04-
01','YYYY-MM-DD'))
,
partition item29 values less than (to_date('1994-05-
01','YYYY-MM-DD'))
,
partition item30 values less than (to_date('1994-06-
01','YYYY-MM-DD'))
,
```

```
partition item31 values less than (to_date('1994-07-
01','YYYY-MM-DD'))
,
partition item32 values less than (to_date('1994-08-
01','YYYY-MM-DD'))
,
partition item33 values less than (to_date('1994-09-
01','YYYY-MM-DD'))
,
partition item34 values less than (to_date('1994-10-
01','YYYY-MM-DD'))
,
partition item35 values less than (to_date('1994-11-
01','YYYY-MM-DD'))
,
partition item36 values less than (to_date('1994-12-
01','YYYY-MM-DD'))
,
partition item37 values less than (to_date('1995-01-
01','YYYY-MM-DD'))
,
partition item38 values less than (to_date('1995-02-
01','YYYY-MM-DD'))
,
partition item39 values less than (to_date('1995-03-
01','YYYY-MM-DD'))
,
partition item40 values less than (to_date('1995-04-
01','YYYY-MM-DD'))
,
partition item41 values less than (to_date('1995-05-
01','YYYY-MM-DD'))
,
partition item42 values less than (to_date('1995-06-
01','YYYY-MM-DD'))
,
partition item43 values less than (to_date('1995-07-
01','YYYY-MM-DD'))
,
partition item44 values less than (to_date('1995-08-
01','YYYY-MM-DD'))
,
partition item45 values less than (to_date('1995-09-
01','YYYY-MM-DD'))
,
partition item46 values less than (to_date('1995-10-
01','YYYY-MM-DD'))
,
partition item47 values less than (to_date('1995-11-
01','YYYY-MM-DD'))
,
partition item48 values less than (to_date('1995-12-
01','YYYY-MM-DD'))
,
partition item49 values less than (to_date('1996-01-
01','YYYY-MM-DD'))
,
partition item50 values less than (to_date('1996-02-
01','YYYY-MM-DD'))
,
partition item51 values less than (to_date('1996-03-
01','YYYY-MM-DD'))
,
partition item52 values less than (to_date('1996-04-
01','YYYY-MM-DD'))
,
partition item53 values less than (to_date('1996-05-
01','YYYY-MM-DD'))
,
partition item54 values less than (to_date('1996-06-
01','YYYY-MM-DD'))
,
partition item55 values less than (to_date('1996-07-
01','YYYY-MM-DD'))
,
partition item56 values less than (to_date('1996-08-
01','YYYY-MM-DD'))
,
partition item57 values less than (to_date('1996-09-
01','YYYY-MM-DD'))
,
partition item58 values less than (to_date('1996-10-
01','YYYY-MM-DD'))
,
partition item59 values less than (to_date('1996-11-
01','YYYY-MM-DD'))
,
partition item60 values less than (to_date('1996-12-
01','YYYY-MM-DD'))
,
partition item61 values less than (to_date('1997-01-
01','YYYY-MM-DD'))
,
partition item62 values less than (to_date('1997-02-
01','YYYY-MM-DD'))
,
partition item63 values less than (to_date('1997-03-
01','YYYY-MM-DD'))
,
partition item64 values less than (to_date('1997-04-
01','YYYY-MM-DD'))
,
partition item65 values less than (to_date('1997-05-
01','YYYY-MM-DD'))
,
partition item66 values less than (to_date('1997-06-
01','YYYY-MM-DD'))
,
partition item67 values less than (to_date('1997-07-
01','YYYY-MM-DD'))
,
partition item68 values less than (to_date('1997-08-
01','YYYY-MM-DD'))
,
```

```
partition item69 values less than (to_date('1997-09-
01','YYYY-MM-DD'))
,
partition item70 values less than (to_date('1997-10-
01','YYYY-MM-DD'))
,
partition item71 values less than (to_date('1997-11-
01','YYYY-MM-DD'))
,
partition item72 values less than (to_date('1997-12-
01','YYYY-MM-DD'))
,
partition item73 values less than (to_date('1998-01-
01','YYYY-MM-DD'))
,
partition item74 values less than (to_date('1998-02-
01','YYYY-MM-DD'))
,
partition item75 values less than (to_date('1998-03-
01','YYYY-MM-DD'))
,
partition item76 values less than (to_date('1998-04-
01','YYYY-MM-DD'))
,
partition item77 values less than (to_date('1998-05-
01','YYYY-MM-DD'))
,
partition item78 values less than (to_date('1998-06-
01','YYYY-MM-DD'))
,
partition item79 values less than (to_date('1998-07-
01','YYYY-MM-DD'))
,
partition item80 values less than (to_date('1998-08-
01','YYYY-MM-DD'))
,
partition item81 values less than (to_date('1998-09-
01','YYYY-MM-DD'))
,
partition item82 values less than (to_date('1998-10-
01','YYYY-MM-DD'))
,
partition item83 values less than (to_date('1998-11-
01','YYYY-MM-DD'))
,
partition item84 values less than (MAXVALUE)
)
as select * from l_et
;
}
*sql
{
connect tpcd/tpcd
set timing on
set echo on

rem drop table orders;
create table orders(
    o_orderdate         ,
    o_orderkey          NOT NULL,
    o_custkey           NOT NULL,
    o_orderpriority     ,
    o_shippriority      ,
    o_clerk             ,
    o_orderstatus       ,
    o_totalprice        ,
    o_comment
)
pctfree 1
storage(freelists 16 initial 5400k next 5400k pctincrease 0
minextents 6 maxextents unlimited)
nologging
initrans 10
parallel 8
partition by range (o_orderdate)
subpartition by hash (o_custkey)
subpartitions 8
(
partition ord1 values less than (to_date('1992-01-
01','YYYY-MM-DD'))
,
partition ord2 values less than (to_date('1992-02-
01','YYYY-MM-DD'))
,
partition ord3 values less than (to_date('1992-03-
01','YYYY-MM-DD'))
,
partition ord4 values less than (to_date('1992-04-
01','YYYY-MM-DD'))
,
partition ord5 values less than (to_date('1992-05-
01','YYYY-MM-DD'))
,
partition ord6 values less than (to_date('1992-06-
01','YYYY-MM-DD'))
,
partition ord7 values less than (to_date('1992-07-
01','YYYY-MM-DD'))
,
partition ord8 values less than (to_date('1992-08-
01','YYYY-MM-DD'))
,
partition ord9 values less than (to_date('1992-09-
01','YYYY-MM-DD'))
,
partition ord10 values less than (to_date('1992-10-
01','YYYY-MM-DD'))
,
partition ord11 values less than (to_date('1992-11-
01','YYYY-MM-DD'))
,
partition ord12 values less than (to_date('1992-12-
01','YYYY-MM-DD'))
,
partition ord13 values less than (to_date('1993-01-
01','YYYY-MM-DD'))
```

```
,
partition ord14 values less than (to_date('1993-02-
01','YYYY-MM-DD'))
,
partition ord15 values less than (to_date('1993-03-
01','YYYY-MM-DD'))
,
partition ord16 values less than (to_date('1993-04-
01','YYYY-MM-DD'))
,
partition ord17 values less than (to_date('1993-05-
01','YYYY-MM-DD'))
,
partition ord18 values less than (to_date('1993-06-
01','YYYY-MM-DD'))
,
partition ord19 values less than (to_date('1993-07-
01','YYYY-MM-DD'))
,
partition ord20 values less than (to_date('1993-08-
01','YYYY-MM-DD'))
,
partition ord21 values less than (to_date('1993-09-
01','YYYY-MM-DD'))
,
partition ord22 values less than (to_date('1993-10-
01','YYYY-MM-DD'))
,
partition ord23 values less than (to_date('1993-11-
01','YYYY-MM-DD'))
,
partition ord24 values less than (to_date('1993-12-
01','YYYY-MM-DD'))
,
partition ord25 values less than (to_date('1994-01-
01','YYYY-MM-DD'))
,
partition ord26 values less than (to_date('1994-02-
01','YYYY-MM-DD'))
,
partition ord27 values less than (to_date('1994-03-
01','YYYY-MM-DD'))
,
partition ord28 values less than (to_date('1994-04-
01','YYYY-MM-DD'))
,
partition ord29 values less than (to_date('1994-05-
01','YYYY-MM-DD'))
,
partition ord30 values less than (to_date('1994-06-
01','YYYY-MM-DD'))
,
partition ord31 values less than (to_date('1994-07-
01','YYYY-MM-DD'))
,
partition ord32 values less than (to_date('1994-08-
01','YYYY-MM-DD'))
,
partition ord33 values less than (to_date('1994-09-
01','YYYY-MM-DD'))
,
partition ord34 values less than (to_date('1994-10-
01','YYYY-MM-DD'))
,
partition ord35 values less than (to_date('1994-11-
01','YYYY-MM-DD'))
,
partition ord36 values less than (to_date('1994-12-
01','YYYY-MM-DD'))
,
partition ord37 values less than (to_date('1995-01-
01','YYYY-MM-DD'))
,
partition ord38 values less than (to_date('1995-02-
01','YYYY-MM-DD'))
,
partition ord39 values less than (to_date('1995-03-
01','YYYY-MM-DD'))
,
partition ord40 values less than (to_date('1995-04-
01','YYYY-MM-DD'))
,
partition ord41 values less than (to_date('1995-05-
01','YYYY-MM-DD'))
,
partition ord42 values less than (to_date('1995-06-
01','YYYY-MM-DD'))
,
partition ord43 values less than (to_date('1995-07-
01','YYYY-MM-DD'))
,
partition ord44 values less than (to_date('1995-08-
01','YYYY-MM-DD'))
,
partition ord45 values less than (to_date('1995-09-
01','YYYY-MM-DD'))
,
partition ord46 values less than (to_date('1995-10-
01','YYYY-MM-DD'))
,
partition ord47 values less than (to_date('1995-11-
01','YYYY-MM-DD'))
,
partition ord48 values less than (to_date('1995-12-
01','YYYY-MM-DD'))
,
partition ord49 values less than (to_date('1996-01-
01','YYYY-MM-DD'))
,
partition ord50 values less than (to_date('1996-02-
01','YYYY-MM-DD'))
,
partition ord51 values less than (to_date('1996-03-
01','YYYY-MM-DD'))
,
```

```
partition ord52 values less than (to_date('1996-04-
01','YYYY-MM-DD'))
,
partition ord53 values less than (to_date('1996-05-
01','YYYY-MM-DD'))
,
partition ord54 values less than (to_date('1996-06-
01','YYYY-MM-DD'))
,
partition ord55 values less than (to_date('1996-07-
01','YYYY-MM-DD'))
,
partition ord56 values less than (to_date('1996-08-
01','YYYY-MM-DD'))
,
partition ord57 values less than (to_date('1996-09-
01','YYYY-MM-DD'))
,
partition ord58 values less than (to_date('1996-10-
01','YYYY-MM-DD'))
,
partition ord59 values less than (to_date('1996-11-
01','YYYY-MM-DD'))
,
partition ord60 values less than (to_date('1996-12-
01','YYYY-MM-DD'))
,
partition ord61 values less than (to_date('1997-01-
01','YYYY-MM-DD'))
,
partition ord62 values less than (to_date('1997-02-
01','YYYY-MM-DD'))
,
partition ord63 values less than (to_date('1997-03-
01','YYYY-MM-DD'))
,
partition ord64 values less than (to_date('1997-04-
01','YYYY-MM-DD'))
,
partition ord65 values less than (to_date('1997-05-
01','YYYY-MM-DD'))
,
partition ord66 values less than (to_date('1997-06-
01','YYYY-MM-DD'))
,
partition ord67 values less than (to_date('1997-07-
01','YYYY-MM-DD'))
,
partition ord68 values less than (to_date('1997-08-
01','YYYY-MM-DD'))
,
partition ord69 values less than (to_date('1997-09-
01','YYYY-MM-DD'))
,
partition ord70 values less than (to_date('1997-10-
01','YYYY-MM-DD'))
,
partition ord71 values less than (to_date('1997-11-
01','YYYY-MM-DD'))
,
partition ord72 values less than (to_date('1997-12-
01','YYYY-MM-DD'))
,
partition ord73 values less than (to_date('1998-01-
01','YYYY-MM-DD'))
,
partition ord74 values less than (to_date('1998-02-
01','YYYY-MM-DD'))
,
partition ord75 values less than (to_date('1998-03-
01','YYYY-MM-DD'))
,
partition ord76 values less than (to_date('1998-04-
01','YYYY-MM-DD'))
,
partition ord77 values less than (to_date('1998-05-
01','YYYY-MM-DD'))
,
partition ord78 values less than (to_date('1998-06-
01','YYYY-MM-DD'))
,
partition ord79 values less than (to_date('1998-07-
01','YYYY-MM-DD'))
,
partition ord80 values less than (to_date('1998-08-
01','YYYY-MM-DD'))
,
partition ord81 values less than (to_date('1998-09-
01','YYYY-MM-DD'))
,
partition ord82 values less than (to_date('1998-10-
01','YYYY-MM-DD'))
,
partition ord83 values less than (to_date('1998-11-
01','YYYY-MM-DD'))
,
partition ord84 values less than (MAXVALUE)
)
as select * from o_et
;
}
*sql
{
connect tpcd/tpcd
set timing on
set echo on

rem drop table partsupp;
create table partsupp(
   ps_partkey          NOT NULL,
   ps_suppkey          NOT NULL,
   ps_supplycost       NOT NULL,
   ps_availqty         ,
   ps_comment
)
```

```
pctfree 0
storage(freelists 16 initial 250512k next 250512k
pctincrease 0 minextents 6 maxextents unlimited)
nologging
initrans 10
parallel 8
partition by hash (ps_partkey)
partitions 8
as select * from ps_et
;
}
*sql
{
connect tpcd/tpcd
set timing on
set echo on

rem drop table parts;
create table parts(
   p_partkey          NOT NULL,
   p_type             ,
   p_size             ,
   p_brand            ,
   p_name             ,
   p_container        ,
   p_mfgr             ,
   p_retailprice      ,
   p_comment
)
pctfree 0
storage(freelists 16 initial 64528k next 64528k
pctincrease 0 minextents 6 maxextents unlimited)
nologging
initrans 10
parallel 8
partition by hash (p_partkey)
partitions 8
as select * from p_et
;
}
*sql
{
connect tpcd/tpcd
set timing on
set echo on

rem drop table customer;
create table customer(
   c_custkey          NOT NULL,
   c_mktsegment       ,
   c_nationkey        ,
   c_name             ,
   c_address          ,
   c_phone            ,
   c_acctbal          ,
   c_comment
)
pctfree 0
storage(freelists 16 initial 51704k next 51704k
pctincrease 0 minextents 6 maxextents unlimited)
nologging
initrans 10
parallel 8
partition by hash (c_custkey)
partitions 8
as select * from c_et
;
}
*sql
{
connect tpcd/tpcd
set timing on
set echo on

rem drop table supplier;
create table supplier(
   s_suppkey          NOT NULL,
   s_nationkey        ,
   s_comment          ,
   s_name             ,
   s_address          ,
   s_phone            ,
   s_acctbal
)
pctfree 0
storage(freelists 16 initial 3152k next 3152k pctincrease 0
minextents 6 maxextents unlimited)
nologging
initrans 10
parallel 8
partition by hash (s_suppkey)
partitions 8
as select * from s_et
;
}
*sql
{
connect tpcd/tpcd
set timing on
set echo on

rem drop table nation;
create table nation(
   n_nationkey        NOT NULL,
   n_name             ,
   n_regionkey        ,
   n_comment
)
storage(freelists 16  initial 16k next 16k pctincrease 0
minextents 1 maxextents unlimited)
nologging
initrans 10
as select * from n_et
;
}
*sql
```

```
{
connect tpcd/tpcd
set timing on
set echo on

rem drop table region;
create table region(
   r_regionkey         ,
   r_name              ,
   r_comment
)
storage(freelists 16  initial 16k next 16k pctincrease 0
minextents 1 maxextents unlimited)
nologging
initrans 10
as select * from r_et
;
}
*sql
{
connect tpcd/tpcd
set timing on
set echo on

drop table l_et;
drop table o_et;
drop table ps_et;
drop table p_et;
drop table s_et;
drop table c_et;
drop table n_et;
drop table r_et;
}
*wait
*bgoff
%e-dapop
*time{Database population end}
%b-scuvo
*bgon=1
################################################
################################
# Schema Creation Phase - Views ONLY (no datafiles)
################################################
################################
#First create unique indexes - pk - for fast refresh
*time{Schema create start - unique index pk}
*sql
{
connect tpcd/tpcd
set timing on
set echo on

rem drop index i_l_okey_lstatus ;
create unique index i_l_okey_lstatus
on lineitem (l_orderkey, l_linenumber)
pctfree 2
storage(freelists 16  initial 416320k next 416320k
pctincrease 0 minextents 1 maxextents unlimited)
nologging
initrans 10
compute statistics
parallel 8
;
}
*sql
{
connect tpcd/tpcd
set timing on
set echo on

rem drop index i_ps_partkey_suppkey;
create unique index i_ps_partkey_suppkey
on partsupp (ps_partkey,ps_suppkey)
pctfree 0
storage(freelists 16  initial 55920k next 55920k
pctincrease 0 minextents 1 maxextents unlimited)
nologging
initrans 10
compute statistics
parallel 8
;
}
*sql
{
connect tpcd/tpcd
set timing on
set echo on

rem drop index i_o_orderkey;
create unique index i_o_orderkey
on orders (o_orderkey)
pctfree 2
storage(freelists 16  initial 90520k next 90520k
pctincrease 0 minextents 1 maxextents unlimited)
nologging
initrans 10
compute statistics
parallel 8
;
}
*sql
{
connect tpcd/tpcd
set timing on
set echo on

rem drop index i_p_partkey;
create unique index i_p_partkey
on parts (p_partkey)
pctfree 0
storage(freelists 16  initial 11192k next 11192k
pctincrease 0 minextents 1 maxextents unlimited)
nologging
initrans 10
compute statistics
parallel 8
```

```
;
}
*sql
{
connect tpcd/tpcd
set timing on
set echo on

rem drop index i_c_custkey;
create unique index i_c_custkey
on customer (c_custkey)
pctfree 0
storage(freelists 16  initial 8384k next 8384k pctincrease
0 minextents 1 maxextents unlimited)
nologging
initrans 10
compute statistics
parallel 8
;
}
*sql
{
connect tpcd/tpcd
set timing on
set echo on

rem drop index i_s_suppkey;
create unique index i_s_suppkey
on supplier (s_suppkey)
pctfree 0
storage(freelists 16  initial 592k next 592k pctincrease 0
minextents 1 maxextents unlimited)
nologging
initrans 10
compute statistics
parallel 8
;
}
*sql
{
connect tpcd/tpcd
set timing on
set echo on

rem drop index i_n_nationkey;
create unique index i_n_nationkey
on nation (n_nationkey)
pctfree 0
storage(freelists 16  initial 16k next 16k pctincrease 0
minextents 1 maxextents unlimited)
nologging
initrans 10
compute statistics
;
}
*sql
{
connect tpcd/tpcd
set timing on
set echo on

rem drop index i_r_regionkey;
create unique index i_r_regionkey
on region (r_regionkey)
pctfree 0
storage(freelists 16  initial 16k next 16k pctincrease 0
minextents 1 maxextents unlimited)
nologging
initrans 10
compute statistics
;
}
*sql
{
connect tpcd/tpcd
set timing on
set echo on

alter table nation add primary key (n_nationkey) using
index;
alter table region add primary key (r_regionkey) using
index;
alter table supplier add primary key (s_suppkey) using
index;
alter table parts add primary key (p_partkey) using index;
alter table partsupp add primary key (ps_partkey,
ps_suppkey) using index;
alter table customer add primary key (c_custkey) using
index;
alter table orders add primary key (o_orderkey) using
index;
alter table lineitem add primary key (l_orderkey,
l_linenumber) using index;
}
# First create the materialized log for the base tables
*sql
{
connect tpcd/tpcd
set timing on
set echo on

rem drop materialized view log on lineitem;
create materialized view log on lineitem
storage(initial 432k next 432k pctincrease 0 minextents 6
maxextents unlimited)
parallel 8
partition by range (l_shipdate)
(
partition item1 values less than (to_date('1992-01-
01','YYYY-MM-DD'))
,
partition item2 values less than (to_date('1992-02-
01','YYYY-MM-DD'))
,
partition item3 values less than (to_date('1992-03-
01','YYYY-MM-DD'))
```

```
,
partition item4 values less than (to_date('1992-04-
01','YYYY-MM-DD'))
,
partition item5 values less than (to_date('1992-05-
01','YYYY-MM-DD'))
,
partition item6 values less than (to_date('1992-06-
01','YYYY-MM-DD'))
,
partition item7 values less than (to_date('1992-07-
01','YYYY-MM-DD'))
,
partition item8 values less than (to_date('1992-08-
01','YYYY-MM-DD'))
,
partition item9 values less than (to_date('1992-09-
01','YYYY-MM-DD'))
,
partition item10 values less than (to_date('1992-10-
01','YYYY-MM-DD'))
,
partition item11 values less than (to_date('1992-11-
01','YYYY-MM-DD'))
,
partition item12 values less than (to_date('1992-12-
01','YYYY-MM-DD'))
,
partition item13 values less than (to_date('1993-01-
01','YYYY-MM-DD'))
,
partition item14 values less than (to_date('1993-02-
01','YYYY-MM-DD'))
,
partition item15 values less than (to_date('1993-03-
01','YYYY-MM-DD'))
,
partition item16 values less than (to_date('1993-04-
01','YYYY-MM-DD'))
,
partition item17 values less than (to_date('1993-05-
01','YYYY-MM-DD'))
,
partition item18 values less than (to_date('1993-06-
01','YYYY-MM-DD'))
,
partition item19 values less than (to_date('1993-07-
01','YYYY-MM-DD'))
,
partition item20 values less than (to_date('1993-08-
01','YYYY-MM-DD'))
,
partition item21 values less than (to_date('1993-09-
01','YYYY-MM-DD'))
,
partition item22 values less than (to_date('1993-10-
01','YYYY-MM-DD'))
,
partition item23 values less than (to_date('1993-11-
01','YYYY-MM-DD'))
,
partition item24 values less than (to_date('1993-12-
01','YYYY-MM-DD'))
,
partition item25 values less than (to_date('1994-01-
01','YYYY-MM-DD'))
,
partition item26 values less than (to_date('1994-02-
01','YYYY-MM-DD'))
,
partition item27 values less than (to_date('1994-03-
01','YYYY-MM-DD'))
,
partition item28 values less than (to_date('1994-04-
01','YYYY-MM-DD'))
,
partition item29 values less than (to_date('1994-05-
01','YYYY-MM-DD'))
,
partition item30 values less than (to_date('1994-06-
01','YYYY-MM-DD'))
,
partition item31 values less than (to_date('1994-07-
01','YYYY-MM-DD'))
,
partition item32 values less than (to_date('1994-08-
01','YYYY-MM-DD'))
,
partition item33 values less than (to_date('1994-09-
01','YYYY-MM-DD'))
,
partition item34 values less than (to_date('1994-10-
01','YYYY-MM-DD'))
,
partition item35 values less than (to_date('1994-11-
01','YYYY-MM-DD'))
,
partition item36 values less than (to_date('1994-12-
01','YYYY-MM-DD'))
,
partition item37 values less than (to_date('1995-01-
01','YYYY-MM-DD'))
,
partition item38 values less than (to_date('1995-02-
01','YYYY-MM-DD'))
,
partition item39 values less than (to_date('1995-03-
01','YYYY-MM-DD'))
,
partition item40 values less than (to_date('1995-04-
01','YYYY-MM-DD'))
,
partition item41 values less than (to_date('1995-05-
01','YYYY-MM-DD'))
,
```

```
partition item42 values less than (to_date('1995-06-
01','YYYY-MM-DD'))
,
partition item43 values less than (to_date('1995-07-
01','YYYY-MM-DD'))
,
partition item44 values less than (to_date('1995-08-
01','YYYY-MM-DD'))
,
partition item45 values less than (to_date('1995-09-
01','YYYY-MM-DD'))
,
partition item46 values less than (to_date('1995-10-
01','YYYY-MM-DD'))
,
partition item47 values less than (to_date('1995-11-
01','YYYY-MM-DD'))
,
partition item48 values less than (to_date('1995-12-
01','YYYY-MM-DD'))
,
partition item49 values less than (to_date('1996-01-
01','YYYY-MM-DD'))
,
partition item50 values less than (to_date('1996-02-
01','YYYY-MM-DD'))
,
partition item51 values less than (to_date('1996-03-
01','YYYY-MM-DD'))
,
partition item52 values less than (to_date('1996-04-
01','YYYY-MM-DD'))
,
partition item53 values less than (to_date('1996-05-
01','YYYY-MM-DD'))
,
partition item54 values less than (to_date('1996-06-
01','YYYY-MM-DD'))
,
partition item55 values less than (to_date('1996-07-
01','YYYY-MM-DD'))
,
partition item56 values less than (to_date('1996-08-
01','YYYY-MM-DD'))
,
partition item57 values less than (to_date('1996-09-
01','YYYY-MM-DD'))
,
partition item58 values less than (to_date('1996-10-
01','YYYY-MM-DD'))
,
partition item59 values less than (to_date('1996-11-
01','YYYY-MM-DD'))
,
partition item60 values less than (to_date('1996-12-
01','YYYY-MM-DD'))
,
partition item61 values less than (to_date('1997-01-
01','YYYY-MM-DD'))
,
partition item62 values less than (to_date('1997-02-
01','YYYY-MM-DD'))
,
partition item63 values less than (to_date('1997-03-
01','YYYY-MM-DD'))
,
partition item64 values less than (to_date('1997-04-
01','YYYY-MM-DD'))
,
partition item65 values less than (to_date('1997-05-
01','YYYY-MM-DD'))
,
partition item66 values less than (to_date('1997-06-
01','YYYY-MM-DD'))
,
partition item67 values less than (to_date('1997-07-
01','YYYY-MM-DD'))
,
partition item68 values less than (to_date('1997-08-
01','YYYY-MM-DD'))
,
partition item69 values less than (to_date('1997-09-
01','YYYY-MM-DD'))
,
partition item70 values less than (to_date('1997-10-
01','YYYY-MM-DD'))
,
partition item71 values less than (to_date('1997-11-
01','YYYY-MM-DD'))
,
partition item72 values less than (to_date('1997-12-
01','YYYY-MM-DD'))
,
partition item73 values less than (to_date('1998-01-
01','YYYY-MM-DD'))
,
partition item74 values less than (to_date('1998-02-
01','YYYY-MM-DD'))
,
partition item75 values less than (to_date('1998-03-
01','YYYY-MM-DD'))
,
partition item76 values less than (to_date('1998-04-
01','YYYY-MM-DD'))
,
partition item77 values less than (to_date('1998-05-
01','YYYY-MM-DD'))
,
partition item78 values less than (to_date('1998-06-
01','YYYY-MM-DD'))
,
partition item79 values less than (to_date('1998-07-
01','YYYY-MM-DD'))
,
```

```
partition item80 values less than (to_date('1998-08-
01','YYYY-MM-DD'))
,
partition item81 values less than (to_date('1998-09-
01','YYYY-MM-DD'))
,
partition item82 values less than (to_date('1998-10-
01','YYYY-MM-DD'))
,
partition item83 values less than (to_date('1998-11-
01','YYYY-MM-DD'))
,
partition item84 values less than (MAXVALUE)
)
with rowid,sequence
(
   l_shipdate          ,
   l_returnflag        ,
   l_linestatus        ,
   l_quantity          ,
   l_extendedprice     ,
   l_discount          ,
   l_tax               ,
   l_suppkey           ,
   l_partkey           ,
   l_orderkey                  ,
   l_shipmode                  ,
   l_receiptdate     ,
   l_commitdate
)
including new values
;
rem drop materialized view log on orders;
create materialized view log on orders
storage(initial 80k next 80k pctincrease 0 minextents 6
maxextents unlimited)
parallel 8
partition by range (o_orderdate)
(
partition ord1 values less than (to_date('1992-01-
01','YYYY-MM-DD'))
,
partition ord2 values less than (to_date('1992-02-
01','YYYY-MM-DD'))
,
partition ord3 values less than (to_date('1992-03-
01','YYYY-MM-DD'))
,
partition ord4 values less than (to_date('1992-04-
01','YYYY-MM-DD'))
,
partition ord5 values less than (to_date('1992-05-
01','YYYY-MM-DD'))
,
partition ord6 values less than (to_date('1992-06-
01','YYYY-MM-DD'))
,
partition ord7 values less than (to_date('1992-07-
01','YYYY-MM-DD'))
,
partition ord8 values less than (to_date('1992-08-
01','YYYY-MM-DD'))
,
partition ord9 values less than (to_date('1992-09-
01','YYYY-MM-DD'))
,
partition ord10 values less than (to_date('1992-10-
01','YYYY-MM-DD'))
,
partition ord11 values less than (to_date('1992-11-
01','YYYY-MM-DD'))
,
partition ord12 values less than (to_date('1992-12-
01','YYYY-MM-DD'))
,
partition ord13 values less than (to_date('1993-01-
01','YYYY-MM-DD'))
,
partition ord14 values less than (to_date('1993-02-
01','YYYY-MM-DD'))
,
partition ord15 values less than (to_date('1993-03-
01','YYYY-MM-DD'))
,
partition ord16 values less than (to_date('1993-04-
01','YYYY-MM-DD'))
,
partition ord17 values less than (to_date('1993-05-
01','YYYY-MM-DD'))
,
partition ord18 values less than (to_date('1993-06-
01','YYYY-MM-DD'))
,
partition ord19 values less than (to_date('1993-07-
01','YYYY-MM-DD'))
,
partition ord20 values less than (to_date('1993-08-
01','YYYY-MM-DD'))
,
partition ord21 values less than (to_date('1993-09-
01','YYYY-MM-DD'))
,
partition ord22 values less than (to_date('1993-10-
01','YYYY-MM-DD'))
,
partition ord23 values less than (to_date('1993-11-
01','YYYY-MM-DD'))
,
partition ord24 values less than (to_date('1993-12-
01','YYYY-MM-DD'))
,
partition ord25 values less than (to_date('1994-01-
01','YYYY-MM-DD'))
,
```

```
partition ord26 values less than (to_date('1994-02-
01','YYYY-MM-DD'))
,
partition ord27 values less than (to_date('1994-03-
01','YYYY-MM-DD'))
,
partition ord28 values less than (to_date('1994-04-
01','YYYY-MM-DD'))
,
partition ord29 values less than (to_date('1994-05-
01','YYYY-MM-DD'))
,
partition ord30 values less than (to_date('1994-06-
01','YYYY-MM-DD'))
,
partition ord31 values less than (to_date('1994-07-
01','YYYY-MM-DD'))
,
partition ord32 values less than (to_date('1994-08-
01','YYYY-MM-DD'))
,
partition ord33 values less than (to_date('1994-09-
01','YYYY-MM-DD'))
,
partition ord34 values less than (to_date('1994-10-
01','YYYY-MM-DD'))
,
partition ord35 values less than (to_date('1994-11-
01','YYYY-MM-DD'))
,
partition ord36 values less than (to_date('1994-12-
01','YYYY-MM-DD'))
,
partition ord37 values less than (to_date('1995-01-
01','YYYY-MM-DD'))
,
partition ord38 values less than (to_date('1995-02-
01','YYYY-MM-DD'))
,
partition ord39 values less than (to_date('1995-03-
01','YYYY-MM-DD'))
,
partition ord40 values less than (to_date('1995-04-
01','YYYY-MM-DD'))
,
partition ord41 values less than (to_date('1995-05-
01','YYYY-MM-DD'))
,
partition ord42 values less than (to_date('1995-06-
01','YYYY-MM-DD'))
,
partition ord43 values less than (to_date('1995-07-
01','YYYY-MM-DD'))
,
partition ord44 values less than (to_date('1995-08-
01','YYYY-MM-DD'))
,
partition ord45 values less than (to_date('1995-09-
01','YYYY-MM-DD'))
,
partition ord46 values less than (to_date('1995-10-
01','YYYY-MM-DD'))
,
partition ord47 values less than (to_date('1995-11-
01','YYYY-MM-DD'))
,
partition ord48 values less than (to_date('1995-12-
01','YYYY-MM-DD'))
,
partition ord49 values less than (to_date('1996-01-
01','YYYY-MM-DD'))
,
partition ord50 values less than (to_date('1996-02-
01','YYYY-MM-DD'))
,
partition ord51 values less than (to_date('1996-03-
01','YYYY-MM-DD'))
,
partition ord52 values less than (to_date('1996-04-
01','YYYY-MM-DD'))
,
partition ord53 values less than (to_date('1996-05-
01','YYYY-MM-DD'))
,
partition ord54 values less than (to_date('1996-06-
01','YYYY-MM-DD'))
,
partition ord55 values less than (to_date('1996-07-
01','YYYY-MM-DD'))
,
partition ord56 values less than (to_date('1996-08-
01','YYYY-MM-DD'))
,
partition ord57 values less than (to_date('1996-09-
01','YYYY-MM-DD'))
,
partition ord58 values less than (to_date('1996-10-
01','YYYY-MM-DD'))
,
partition ord59 values less than (to_date('1996-11-
01','YYYY-MM-DD'))
,
partition ord60 values less than (to_date('1996-12-
01','YYYY-MM-DD'))
,
partition ord61 values less than (to_date('1997-01-
01','YYYY-MM-DD'))
,
partition ord62 values less than (to_date('1997-02-
01','YYYY-MM-DD'))
,
partition ord63 values less than (to_date('1997-03-
01','YYYY-MM-DD'))
,
```

```
partition ord64 values less than (to_date('1997-04-
01','YYYY-MM-DD'))
,
partition ord65 values less than (to_date('1997-05-
01','YYYY-MM-DD'))
,
partition ord66 values less than (to_date('1997-06-
01','YYYY-MM-DD'))
,
partition ord67 values less than (to_date('1997-07-
01','YYYY-MM-DD'))
,
partition ord68 values less than (to_date('1997-08-
01','YYYY-MM-DD'))
,
partition ord69 values less than (to_date('1997-09-
01','YYYY-MM-DD'))
,
partition ord70 values less than (to_date('1997-10-
01','YYYY-MM-DD'))
,
partition ord71 values less than (to_date('1997-11-
01','YYYY-MM-DD'))
,
partition ord72 values less than (to_date('1997-12-
01','YYYY-MM-DD'))
,
partition ord73 values less than (to_date('1998-01-
01','YYYY-MM-DD'))
,
partition ord74 values less than (to_date('1998-02-
01','YYYY-MM-DD'))
,
partition ord75 values less than (to_date('1998-03-
01','YYYY-MM-DD'))
,
partition ord76 values less than (to_date('1998-04-
01','YYYY-MM-DD'))
,
partition ord77 values less than (to_date('1998-05-
01','YYYY-MM-DD'))
,
partition ord78 values less than (to_date('1998-06-
01','YYYY-MM-DD'))
,
partition ord79 values less than (to_date('1998-07-
01','YYYY-MM-DD'))
,
partition ord80 values less than (to_date('1998-08-
01','YYYY-MM-DD'))
,
partition ord81 values less than (to_date('1998-09-
01','YYYY-MM-DD'))
,
partition ord82 values less than (to_date('1998-10-
01','YYYY-MM-DD'))
,
partition ord83 values less than (to_date('1998-11-
01','YYYY-MM-DD'))
,
partition ord84 values less than (MAXVALUE)
)
with rowid,sequence
(
   o_orderdate, o_orderkey, o_custkey, o_orderpriority
)
including new values
;
rem drop materialized view log on customer;
create materialized view log on customer
storage(initial 16k next 16k pctincrease 0 minextents 1
maxextents unlimited)
parallel 8
with rowid,sequence
(c_nationkey, c_mktsegment,c_custkey)
including new values
;
rem drop materialized view log on  supplier;
create materialized view log on  supplier
storage(initial 16k next 16k pctincrease 0 minextents 1
maxextents unlimited)
parallel 8
with rowid,sequence
(s_nationkey, s_suppkey)
including new values
;
rem drop materialized view log on partsupp;
create materialized view log on partsupp
storage(initial 16k next 16k pctincrease 0 minextents 1
maxextents unlimited)
parallel 8
with rowid,sequence
including new values
;
rem drop materialized view log on parts;
create materialized view log on parts
storage(initial 16k next 16k pctincrease 0 minextents 1
maxextents unlimited)
parallel 8
with rowid,sequence
(p_partkey)
including new values
;
rem drop materialized view log on nation;
create materialized view log on nation
storage(initial 16k next 16k pctincrease 0 minextents 1
maxextents unlimited)
with rowid,sequence
including new values
;
rem drop materialized view log on region;
create materialized view log on region
storage(initial 16k next 16k pctincrease 0 minextents 1
maxextents unlimited)
with rowid,sequence
```

```
including new values
;
}
*wait
*time{Schema create end - unique index pk}
##############################################
##############################
# Schema Creation Phase - Views ONLY (no datafiles)
##############################################
##############################
# Now I can create the views with the corresponding
materialized logs
*time{Schema create start - views with the corresponding
materialized logs}
*sql
{
connect tpcd/tpcd
set timing on
set echo on

rem drop materialized view ma_lineitem;
create materialized view ma_lineitem
pctfree 2
storage(freelists 16  initial 472k next 472k pctincrease 0
minextents 1 maxextents unlimited)
nologging
initrans 10
using no index
REFRESH fast
ON commit
enable query rewrite
as select
count(*) as count_p_group,
            l_shipdate,
            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,
            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,
            count(l_quantity) as count_qty,
            count(l_extendedprice) as count_price,
            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
set timing on
set echo on

rem drop materialized view ma_l_sdfy;
create materialized view ma_l_sdfy
pctfree 2
storage(freelists 16  initial 184k next 184k pctincrease 0
minextents 1 maxextents unlimited)
nologging
initrans 10
using no index
refresh fast on commit
enable query rewrite
as
select
  l_discount,
  l_quantity,
to_char(l_shipdate,'YYYY') as year,
  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,
to_char(l_shipdate,'YYYY')
;
}
*sql
{
connect tpcd/tpcd
set timing on
set echo on

rem drop materialized view ma_rd;
create materialized view ma_rd
pctfree 2
storage(freelists 16  initial 248k next 248k pctincrease 0
minextents 1 maxextents unlimited)
nologging
initrans 10
using no index
refresh fast on commit
enable query rewrite
   as
   select  /*+ ordered use_hash(orders lineitem) */
     l_shipmode,
     o_orderpriority,
     sum(1) as sum_1,
     sum(0) as sum_0,
     count(*) as count_s,
     count(0) as count_0,
     count(1) as count_1,
```

```
    to_char(l_receiptdate,'YYYY-MM') as l_year_month
   from
    lineitem, orders
   where
   o_orderkey=l_orderkey and
   l_commitdate < l_receiptdate
   and l_shipdate < l_commitdate
   group by
    l_shipmode,
    o_orderpriority,
    to_char(l_receiptdate,'YYYY-MM');
}
*sql
{
connect tpcd/tpcd
set timing on
set echo on

rem drop materialized view mj_locs;
create materialized view mj_locs
pctfree 2
storage(freelists 16 initial 132288k next 132288k
pctincrease 0 minextents 6 maxextents unlimited)
parallel 8
nologging
initrans 10
partition by range (l_shipdate)
subpartition by hash(c_mktsegment)
subpartitions 5
(
partition mj_locs1 values less than (to_date('1992-01-
01','YYYY-MM-DD'))
,
partition mj_locs2 values less than (to_date('1992-02-
01','YYYY-MM-DD'))
,
partition mj_locs3 values less than (to_date('1992-03-
01','YYYY-MM-DD'))
,
partition mj_locs4 values less than (to_date('1992-04-
01','YYYY-MM-DD'))
,
partition mj_locs5 values less than (to_date('1992-05-
01','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('1992-07-
01','YYYY-MM-DD'))
,
partition mj_locs8 values less than (to_date('1992-08-
01','YYYY-MM-DD'))
,
partition mj_locs9 values less than (to_date('1992-09-
01','YYYY-MM-DD'))
,
partition mj_locs10 values less than (to_date('1992-10-
01','YYYY-MM-DD'))
,
partition mj_locs11 values less than (to_date('1992-11-
01','YYYY-MM-DD'))
,
partition mj_locs12 values less than (to_date('1992-12-
01','YYYY-MM-DD'))
,
partition mj_locs13 values less than (to_date('1993-01-
01','YYYY-MM-DD'))
,
partition mj_locs14 values less than (to_date('1993-02-
01','YYYY-MM-DD'))
,
partition mj_locs15 values less than (to_date('1993-03-
01','YYYY-MM-DD'))
,
partition mj_locs16 values less than (to_date('1993-04-
01','YYYY-MM-DD'))
,
partition mj_locs17 values less than (to_date('1993-05-
01','YYYY-MM-DD'))
,
partition mj_locs18 values less than (to_date('1993-06-
01','YYYY-MM-DD'))
,
partition mj_locs19 values less than (to_date('1993-07-
01','YYYY-MM-DD'))
,
partition mj_locs20 values less than (to_date('1993-08-
01','YYYY-MM-DD'))
,
partition mj_locs21 values less than (to_date('1993-09-
01','YYYY-MM-DD'))
,
partition mj_locs22 values less than (to_date('1993-10-
01','YYYY-MM-DD'))
,
partition mj_locs23 values less than (to_date('1993-11-
01','YYYY-MM-DD'))
,
partition mj_locs24 values less than (to_date('1993-12-
01','YYYY-MM-DD'))
,
partition mj_locs25 values less than (to_date('1994-01-
01','YYYY-MM-DD'))
,
partition mj_locs26 values less than (to_date('1994-02-
01','YYYY-MM-DD'))
,
partition mj_locs27 values less than (to_date('1994-03-
01','YYYY-MM-DD'))
,
partition mj_locs28 values less than (to_date('1994-04-
01','YYYY-MM-DD'))
,
```

```
partition mj_locs29 values less than (to_date('1994-05-
01','YYYY-MM-DD'))
,
partition mj_locs30 values less than (to_date('1994-06-
01','YYYY-MM-DD'))
,
partition mj_locs31 values less than (to_date('1994-07-
01','YYYY-MM-DD'))
,
partition mj_locs32 values less than (to_date('1994-08-
01','YYYY-MM-DD'))
,
partition mj_locs33 values less than (to_date('1994-09-
01','YYYY-MM-DD'))
,
partition mj_locs34 values less than (to_date('1994-10-
01','YYYY-MM-DD'))
,
partition mj_locs35 values less than (to_date('1994-11-
01','YYYY-MM-DD'))
,
partition mj_locs36 values less than (to_date('1994-12-
01','YYYY-MM-DD'))
,
partition mj_locs37 values less than (to_date('1995-01-
01','YYYY-MM-DD'))
,
partition mj_locs38 values less than (to_date('1995-02-
01','YYYY-MM-DD'))
,
partition mj_locs39 values less than (to_date('1995-03-
01','YYYY-MM-DD'))
,
partition mj_locs40 values less than (to_date('1995-04-
01','YYYY-MM-DD'))
,
partition mj_locs41 values less than (to_date('1995-05-
01','YYYY-MM-DD'))
,
partition mj_locs42 values less than (to_date('1995-06-
01','YYYY-MM-DD'))
,
partition mj_locs43 values less than (to_date('1995-07-
01','YYYY-MM-DD'))
,
partition mj_locs44 values less than (to_date('1995-08-
01','YYYY-MM-DD'))
,
partition mj_locs45 values less than (to_date('1995-09-
01','YYYY-MM-DD'))
,
partition mj_locs46 values less than (to_date('1995-10-
01','YYYY-MM-DD'))
,
partition mj_locs47 values less than (to_date('1995-11-
01','YYYY-MM-DD'))
,
partition mj_locs48 values less than (to_date('1995-12-
01','YYYY-MM-DD'))
,
partition mj_locs49 values less than (to_date('1996-01-
01','YYYY-MM-DD'))
,
partition mj_locs50 values less than (to_date('1996-02-
01','YYYY-MM-DD'))
,
partition mj_locs51 values less than (to_date('1996-03-
01','YYYY-MM-DD'))
,
partition mj_locs52 values less than (to_date('1996-04-
01','YYYY-MM-DD'))
,
partition mj_locs53 values less than (to_date('1996-05-
01','YYYY-MM-DD'))
,
partition mj_locs54 values less than (to_date('1996-06-
01','YYYY-MM-DD'))
,
partition mj_locs55 values less than (to_date('1996-07-
01','YYYY-MM-DD'))
,
partition mj_locs56 values less than (to_date('1996-08-
01','YYYY-MM-DD'))
,
partition mj_locs57 values less than (to_date('1996-09-
01','YYYY-MM-DD'))
,
partition mj_locs58 values less than (to_date('1996-10-
01','YYYY-MM-DD'))
,
partition mj_locs59 values less than (to_date('1996-11-
01','YYYY-MM-DD'))
,
partition mj_locs60 values less than (to_date('1996-12-
01','YYYY-MM-DD'))
,
partition mj_locs61 values less than (to_date('1997-01-
01','YYYY-MM-DD'))
,
partition mj_locs62 values less than (to_date('1997-02-
01','YYYY-MM-DD'))
,
partition mj_locs63 values less than (to_date('1997-03-
01','YYYY-MM-DD'))
,
partition mj_locs64 values less than (to_date('1997-04-
01','YYYY-MM-DD'))
,
partition mj_locs65 values less than (to_date('1997-05-
01','YYYY-MM-DD'))
,
partition mj_locs66 values less than (to_date('1997-06-
01','YYYY-MM-DD'))
,
```

```
partition mj_locs67 values less than (to_date('1997-07-
01','YYYY-MM-DD'))
,
partition mj_locs68 values less than (to_date('1997-08-
01','YYYY-MM-DD'))
,
partition mj_locs69 values less than (to_date('1997-09-
01','YYYY-MM-DD'))
,
partition mj_locs70 values less than (to_date('1997-10-
01','YYYY-MM-DD'))
,
partition mj_locs71 values less than (to_date('1997-11-
01','YYYY-MM-DD'))
,
partition mj_locs72 values less than (to_date('1997-12-
01','YYYY-MM-DD'))
,
partition mj_locs73 values less than (to_date('1998-01-
01','YYYY-MM-DD'))
,
partition mj_locs74 values less than (to_date('1998-02-
01','YYYY-MM-DD'))
,
partition mj_locs75 values less than (to_date('1998-03-
01','YYYY-MM-DD'))
,
partition mj_locs76 values less than (to_date('1998-04-
01','YYYY-MM-DD'))
,
partition mj_locs77 values less than (to_date('1998-05-
01','YYYY-MM-DD'))
,
partition mj_locs78 values less than (to_date('1998-06-
01','YYYY-MM-DD'))
,
partition mj_locs79 values less than (to_date('1998-07-
01','YYYY-MM-DD'))
,
partition mj_locs80 values less than (to_date('1998-08-
01','YYYY-MM-DD'))
,
partition mj_locs81 values less than (to_date('1998-09-
01','YYYY-MM-DD'))
,
partition mj_locs82 values less than (to_date('1998-10-
01','YYYY-MM-DD'))
,
partition mj_locs83 values less than (to_date('1998-11-
01','YYYY-MM-DD'))
,
partition mj_locs84 values less than (MAXVALUE)
)
enable row movement
REFRESH fast
ON commit
enable query rewrite
as select
 l_shipdate,
 o_orderdate,
 l_extendedprice,
 l_discount,
 c_custkey,
 s_suppkey,
 c_nationkey,
 s_nationkey,
 decode(orders.rowid, null, 1, 0) as o_ajm,
 decode(customer.rowid, null, 1, 0) as c_ajm,
 decode(supplier.rowid, null, 1, 0) 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,
 decode(parts.rowid, null, 1, 0) 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
 lineitem,
 orders,
 customer,
 supplier,
 parts
WHERE
 l_orderkey=o_orderkey(+)
 AND
                o_custkey=c_custkey(+)
 AND
                l_suppkey=s_suppkey(+)
AND
 l_partkey=p_partkey(+)
;
}
*wait
*sql
{
connect tpcd/tpcd
set timing on
set echo on

rem drop materialized view log on mj_locs;
create materialized view log on mj_locs
storage(initial 752k next 752k pctincrease 0 minextents 6
maxextents unlimited)
```

```
parallel 8
partition by range (l_shipdate)
(
partition mj_locs1 values less than (to_date('1992-01-
01','YYYY-MM-DD'))
,
partition mj_locs2 values less than (to_date('1992-02-
01','YYYY-MM-DD'))
,
partition mj_locs3 values less than (to_date('1992-03-
01','YYYY-MM-DD'))
,
partition mj_locs4 values less than (to_date('1992-04-
01','YYYY-MM-DD'))
,
partition mj_locs5 values less than (to_date('1992-05-
01','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('1992-07-
01','YYYY-MM-DD'))
,
partition mj_locs8 values less than (to_date('1992-08-
01','YYYY-MM-DD'))
,
partition mj_locs9 values less than (to_date('1992-09-
01','YYYY-MM-DD'))
,
partition mj_locs10 values less than (to_date('1992-10-
01','YYYY-MM-DD'))
,
partition mj_locs11 values less than (to_date('1992-11-
01','YYYY-MM-DD'))
,
partition mj_locs12 values less than (to_date('1992-12-
01','YYYY-MM-DD'))
,
partition mj_locs13 values less than (to_date('1993-01-
01','YYYY-MM-DD'))
,
partition mj_locs14 values less than (to_date('1993-02-
01','YYYY-MM-DD'))
,
partition mj_locs15 values less than (to_date('1993-03-
01','YYYY-MM-DD'))
,
partition mj_locs16 values less than (to_date('1993-04-
01','YYYY-MM-DD'))
,
partition mj_locs17 values less than (to_date('1993-05-
01','YYYY-MM-DD'))
,
partition mj_locs18 values less than (to_date('1993-06-
01','YYYY-MM-DD'))
,
partition mj_locs19 values less than (to_date('1993-07-
01','YYYY-MM-DD'))
,
partition mj_locs20 values less than (to_date('1993-08-
01','YYYY-MM-DD'))
,
partition mj_locs21 values less than (to_date('1993-09-
01','YYYY-MM-DD'))
,
partition mj_locs22 values less than (to_date('1993-10-
01','YYYY-MM-DD'))
,
partition mj_locs23 values less than (to_date('1993-11-
01','YYYY-MM-DD'))
,
partition mj_locs24 values less than (to_date('1993-12-
01','YYYY-MM-DD'))
,
partition mj_locs25 values less than (to_date('1994-01-
01','YYYY-MM-DD'))
,
partition mj_locs26 values less than (to_date('1994-02-
01','YYYY-MM-DD'))
,
partition mj_locs27 values less than (to_date('1994-03-
01','YYYY-MM-DD'))
,
partition mj_locs28 values less than (to_date('1994-04-
01','YYYY-MM-DD'))
,
partition mj_locs29 values less than (to_date('1994-05-
01','YYYY-MM-DD'))
,
partition mj_locs30 values less than (to_date('1994-06-
01','YYYY-MM-DD'))
,
partition mj_locs31 values less than (to_date('1994-07-
01','YYYY-MM-DD'))
,
partition mj_locs32 values less than (to_date('1994-08-
01','YYYY-MM-DD'))
,
partition mj_locs33 values less than (to_date('1994-09-
01','YYYY-MM-DD'))
,
partition mj_locs34 values less than (to_date('1994-10-
01','YYYY-MM-DD'))
,
partition mj_locs35 values less than (to_date('1994-11-
01','YYYY-MM-DD'))
,
partition mj_locs36 values less than (to_date('1994-12-
01','YYYY-MM-DD'))
,
partition mj_locs37 values less than (to_date('1995-01-
01','YYYY-MM-DD'))
,
```

```
partition mj_locs38 values less than (to_date('1995-02-
01','YYYY-MM-DD'))
,
partition mj_locs39 values less than (to_date('1995-03-
01','YYYY-MM-DD'))
,
partition mj_locs40 values less than (to_date('1995-04-
01','YYYY-MM-DD'))
,
partition mj_locs41 values less than (to_date('1995-05-
01','YYYY-MM-DD'))
,
partition mj_locs42 values less than (to_date('1995-06-
01','YYYY-MM-DD'))
,
partition mj_locs43 values less than (to_date('1995-07-
01','YYYY-MM-DD'))
,
partition mj_locs44 values less than (to_date('1995-08-
01','YYYY-MM-DD'))
,
partition mj_locs45 values less than (to_date('1995-09-
01','YYYY-MM-DD'))
,
partition mj_locs46 values less than (to_date('1995-10-
01','YYYY-MM-DD'))
,
partition mj_locs47 values less than (to_date('1995-11-
01','YYYY-MM-DD'))
,
partition mj_locs48 values less than (to_date('1995-12-
01','YYYY-MM-DD'))
,
partition mj_locs49 values less than (to_date('1996-01-
01','YYYY-MM-DD'))
,
partition mj_locs50 values less than (to_date('1996-02-
01','YYYY-MM-DD'))
,
partition mj_locs51 values less than (to_date('1996-03-
01','YYYY-MM-DD'))
,
partition mj_locs52 values less than (to_date('1996-04-
01','YYYY-MM-DD'))
,
partition mj_locs53 values less than (to_date('1996-05-
01','YYYY-MM-DD'))
,
partition mj_locs54 values less than (to_date('1996-06-
01','YYYY-MM-DD'))
,
partition mj_locs55 values less than (to_date('1996-07-
01','YYYY-MM-DD'))
,
partition mj_locs56 values less than (to_date('1996-08-
01','YYYY-MM-DD'))
,
partition mj_locs57 values less than (to_date('1996-09-
01','YYYY-MM-DD'))
,
partition mj_locs58 values less than (to_date('1996-10-
01','YYYY-MM-DD'))
,
partition mj_locs59 values less than (to_date('1996-11-
01','YYYY-MM-DD'))
,
partition mj_locs60 values less than (to_date('1996-12-
01','YYYY-MM-DD'))
,
partition mj_locs61 values less than (to_date('1997-01-
01','YYYY-MM-DD'))
,
partition mj_locs62 values less than (to_date('1997-02-
01','YYYY-MM-DD'))
,
partition mj_locs63 values less than (to_date('1997-03-
01','YYYY-MM-DD'))
,
partition mj_locs64 values less than (to_date('1997-04-
01','YYYY-MM-DD'))
,
partition mj_locs65 values less than (to_date('1997-05-
01','YYYY-MM-DD'))
,
partition mj_locs66 values less than (to_date('1997-06-
01','YYYY-MM-DD'))
,
partition mj_locs67 values less than (to_date('1997-07-
01','YYYY-MM-DD'))
,
partition mj_locs68 values less than (to_date('1997-08-
01','YYYY-MM-DD'))
,
partition mj_locs69 values less than (to_date('1997-09-
01','YYYY-MM-DD'))
,
partition mj_locs70 values less than (to_date('1997-10-
01','YYYY-MM-DD'))
,
partition mj_locs71 values less than (to_date('1997-11-
01','YYYY-MM-DD'))
,
partition mj_locs72 values less than (to_date('1997-12-
01','YYYY-MM-DD'))
,
partition mj_locs73 values less than (to_date('1998-01-
01','YYYY-MM-DD'))
,
partition mj_locs74 values less than (to_date('1998-02-
01','YYYY-MM-DD'))
,
partition mj_locs75 values less than (to_date('1998-03-
01','YYYY-MM-DD'))
,
```

```
partition mj_locs76 values less than (to_date('1998-04-
01','YYYY-MM-DD'))
,
partition mj_locs77 values less than (to_date('1998-05-
01','YYYY-MM-DD'))
,
partition mj_locs78 values less than (to_date('1998-06-
01','YYYY-MM-DD'))
,
partition mj_locs79 values less than (to_date('1998-07-
01','YYYY-MM-DD'))
,
partition mj_locs80 values less than (to_date('1998-08-
01','YYYY-MM-DD'))
,
partition mj_locs81 values less than (to_date('1998-09-
01','YYYY-MM-DD'))
,
partition mj_locs82 values less than (to_date('1998-10-
01','YYYY-MM-DD'))
,
partition mj_locs83 values less than (to_date('1998-11-
01','YYYY-MM-DD'))
,
partition mj_locs84 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
set timing on
set echo on

rem drop materialized view ma_plocs;
create materialized view ma_plocs
pctfree 2
storage(freelists 16  initial 952k next 952k pctincrease 0
minextents 1 maxextents unlimited)
nologging
initrans 10
using no index
REFRESH fast
ON commit
  enable query rewrite
  as
select
  to_char(l_shipdate,'YYYY-MM') as ship_year_m,
  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_locs
group by
  to_char(l_shipdate,'YYYY-MM') , p_type, p_ajm
;
}
*sql
{
connect tpcd/tpcd
set timing on
set echo on

rem drop materialized view ma_locs;
create materialized view ma_locs
pctfree 2
storage(freelists 16  initial 568k next 568k pctincrease 0
minextents 1 maxextents unlimited)
nologging
initrans 10
using no index
REFRESH fast
ON commit
enable query rewrite
as
select
  to_char(l_shipdate,'YYYY') as ship_year,
  to_char(o_orderdate,'YYYY') as order_year,
  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_locs
group by
  to_char(l_shipdate,'YYYY') ,
  to_char(o_orderdate,'YYYY') ,
  c_nationkey,
  s_nationkey,
  o_ajm,
  c_ajm,
  s_ajm
;
}
*sql
{
connect tpcd/tpcd
set timing on
set echo on

rem drop  materialized view mj_ppssrn;
create materialized view mj_ppssrn
pctfree 0
```

```
storage(freelists 16  initial 107064k next 107064k
pctincrease 0 minextents 6 maxextents unlimited)
parallel 8
nologging
initrans 10
partition by range (r_name)
subpartition by hash(p_size)
subpartitions 8
(
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,
  decode(n.rowid, null, 1, 0) as n_ajm,
  p_brand,
  p_mfgr,
  p_name,
  p_size,
  p_type,
  p_partkey,
  decode(p.rowid, null, 1, 0) as p_ajm,
  ps_availqty,
  ps_partkey,
  ps_suppkey,
  ps_supplycost,
  s_suppkey,
  decode(s.rowid, null, 1, 0) as s_ajm,
  decode(r.rowid, null, 1, 0) as r_ajm,
  r_name ,
  s_nationkey,
  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
    parts p,
  supplier s,
  partsupp ps,
 nation n,
 region r
WHERE
   ps_partkey = p_partkey(+)
 AND
  ps_suppkey = s_suppkey(+)
 AND
  s_nationkey = n_nationkey(+)
 AND
  n_regionkey = r_regionkey(+)
;
}
*sql
{
connect tpcd/tpcd
set timing on
set echo on

rem drop materialized view ma_qty;
create materialized view ma_qty
pctfree 2
storage(freelists 16  initial 14336k next 14336k
pctincrease 0 minextents 1 maxextents unlimited)
parallel 8
nologging
initrans 10
partition by range (sum_qty)
(
partition ma_qty1 values less than ('50')
,
partition ma_qty2 values less than ('100')
,
partition ma_qty3 values less than ('150')
,
partition ma_qty4 values less than ('200')
,
partition ma_qty5 values less than ('250')
,
partition ma_qty6 values less than ('300')
,
partition ma_qty7 values less than (MAXVALUE)
)
enable row movement
using no index
REFRESH fast
ON commit
enable query rewrite
as select
count(*) as count_o_group,
       l_orderkey,
       sum(l_quantity) as sum_qty,
       count(l_quantity) as count_qty
FROM
   lineitem
GROUP BY
l_orderkey
;
}
*wait
*time{Schema create end - views with the corresponding
materialized logs}
*bgoff
%e-scuvo
%b-ixcre
```

```
*bgon=1
##############################################
################################
# Index creation Phase
*time{Index creation start}
*sql
{
connect tpcd/tpcd
set timing on
set echo on

rem drop index ij_l_lrid;
create index ij_l_lrid
on mj_locs (l_rowid)
pctfree 2
storage(freelists 16  initial 45056k next 45056k
pctincrease 0 minextents 1 maxextents unlimited)
nologging
initrans 10
compute statistics
parallel 8
;
}
*sql
{
connect tpcd/tpcd
set timing on
set echo on

rem drop index ij_l_orid;
create index ij_l_orid
on mj_locs (o_rowid)
pctfree 2
storage(freelists 16  initial 45056k next 45056k
pctincrease 0 minextents 1 maxextents unlimited)
nologging
initrans 10
compute statistics
parallel 8
;
}
*sql
{
connect tpcd/tpcd
set timing on
set echo on

rem drop index ij_l_sdate;
create index ij_l_sdate
on mj_locs
(o_orderkey,o_orderdate,o_ajm,c_ajm,l_extendedprice,l_
discount,c_mktsegment,l_shipdate
        ,l_orderkey,l_partkey,o_shippriority)
pctfree 2
storage(freelists 16  initial 39312k next 39312k
pctincrease 0 minextents 1 maxextents unlimited)
nologging
initrans 10
compute statistics
parallel 8
local
;
}
*sql
{
connect tpcd/tpcd
set timing on
set echo on

rem drop index ij_l_ptp;
create index ij_l_ptp
on mj_locs (p_type,
        o_orderkey,
        l_extendedprice,
        l_discount,
        o_orderdate,
        o_ajm,
        c_ajm,
        s_ajm,
        p_ajm,
        c_nationkey,
        s_nationkey,
        l_partkey,
        l_suppkey,
        l_quantity)
pctfree 2
storage(freelists 16  initial 68272k next 68272k
pctincrease 0 minextents 1 maxextents unlimited)
initrans 10
nologging
compute statistics
parallel 8
global partition by range (p_type)
(
partition ptp1 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)
)
;
}
*sql
{
connect tpcd/tpcd
set timing on
set echo on

rem drop index ij_l_odate;
create index ij_l_odate
on mj_locs
(o_orderdate,o_orderkey,l_extendedprice,l_discount,o_aj
m,c_ajm,l_commitdate,

l_receiptdate,o_orderpriority,c_custkey,l_returnflag,c_nati
onkey)
pctfree 2
storage(freelists 16  initial 124896k next 124896k
pctincrease 0 minextents 1 maxextents unlimited)
nologging
initrans 10
compute statistics
parallel 8
global partition by range (o_orderdate)
(
partition ij_l_odate1 values less than (to_date('1992-01-
01','YYYY-MM-DD'))
,
partition ij_l_odate2 values less than (to_date('1992-02-
01','YYYY-MM-DD'))
,
partition ij_l_odate3 values less than (to_date('1992-03-
01','YYYY-MM-DD'))
,
```

```
partition ij_l_odate4 values less than (to_date('1992-04-
1','YYYY-MM-DD'))
,
partition ij_l_odate5 values less than (to_date('1992-05-
01','YYYY-MM-DD'))
,
partition ij_l_odate6 values less than (to_date('1992-06-
01','YYYY-MM-DD'))
,
partition ij_l_odate7 values less than (to_date('1992-07-
01','YYYY-MM-DD'))
,
partition ij_l_odate8 values less than (to_date('1992-08-
01','YYYY-MM-DD'))
,
partition ij_l_odate9 values less than (to_date('1992-09-
01','YYYY-MM-DD'))
,
partition ij_l_odate10 values less than (to_date('1992-10-
01','YYYY-MM-DD'))
,
partition ij_l_odate11 values less than (to_date('1992-11-
01','YYYY-MM-DD'))
,
partition ij_l_odate12 values less than (to_date('1992-12-
01','YYYY-MM-DD'))
,
partition ij_l_odate13 values less than (to_date('1993-01-
01','YYYY-MM-DD'))
,
partition ij_l_odate14 values less than (to_date('1993-02-
01','YYYY-MM-DD'))
,
partition ij_l_odate15 values less than (to_date('1993-03-
01','YYYY-MM-DD'))
,
partition ij_l_odate16 values less than (to_date('1993-04-
01','YYYY-MM-DD'))
,
partition ij_l_odate17 values less than (to_date('1993-05-
01','YYYY-MM-DD'))
,
partition ij_l_odate18 values less than (to_date('1993-06-
01','YYYY-MM-DD'))
,
partition ij_l_odate19 values less than (to_date('1993-07-
01','YYYY-MM-DD'))
,
partition ij_l_odate20 values less than (to_date('1993-08-
01','YYYY-MM-DD'))
,
partition ij_l_odate21 values less than (to_date('1993-09-
01','YYYY-MM-DD'))
,
partition ij_l_odate22 values less than (to_date('1993-10-
01','YYYY-MM-DD'))
,
partition ij_l_odate23 values less than (to_date('1993-11-
01','YYYY-MM-DD'))
,
partition ij_l_odate24 values less than (to_date('1993-12-
01','YYYY-MM-DD'))
,
partition ij_l_odate25 values less than (to_date('1994-01-
01','YYYY-MM-DD'))
,
partition ij_l_odate26 values less than (to_date('1994-02-
01','YYYY-MM-DD'))
,
partition ij_l_odate27 values less than (to_date('1994-03-
01','YYYY-MM-DD'))
,
partition ij_l_odate28 values less than (to_date('1994-04-
01','YYYY-MM-DD'))
,
partition ij_l_odate29 values less than (to_date('1994-05-
01','YYYY-MM-DD'))
,
partition ij_l_odate30 values less than (to_date('1994-06-
01','YYYY-MM-DD'))
,
partition ij_l_odate31 values less than (to_date('1994-07-
01','YYYY-MM-DD'))
,
partition ij_l_odate32 values less than (to_date('1994-08-
01','YYYY-MM-DD'))
,
partition ij_l_odate33 values less than (to_date('1994-09-
01','YYYY-MM-DD'))
,
partition ij_l_odate34 values less than (to_date('1994-10-
01','YYYY-MM-DD'))
,
partition ij_l_odate35 values less than (to_date('1994-11-
01','YYYY-MM-DD'))
,
partition ij_l_odate36 values less than (to_date('1994-12-
01','YYYY-MM-DD'))
,
partition ij_l_odate37 values less than (to_date('1995-01-
01','YYYY-MM-DD'))
,
partition ij_l_odate38 values less than (to_date('1995-02-
01','YYYY-MM-DD'))
,
partition ij_l_odate39 values less than (to_date('1995-03-
01','YYYY-MM-DD'))
,
partition ij_l_odate40 values less than (to_date('1995-04-
01','YYYY-MM-DD'))
,
partition ij_l_odate41 values less than (to_date('1995-05-
01','YYYY-MM-DD'))
,
```

```
partition ij_l_odate42 values less than (to_date('1995-06-
01','YYYY-MM-DD'))
,
partition ij_l_odate43 values less than (to_date('1995-07-
01','YYYY-MM-DD'))
,
partition ij_l_odate44 values less than (to_date('1995-08-
01','YYYY-MM-DD'))
,
partition ij_l_odate45 values less than (to_date('1995-09-
01','YYYY-MM-DD'))
,
partition ij_l_odate46 values less than (to_date('1995-10-
01','YYYY-MM-DD'))
,
partition ij_l_odate47 values less than (to_date('1995-11-
01','YYYY-MM-DD'))
,
partition ij_l_odate48 values less than (to_date('1995-12-
01','YYYY-MM-DD'))
,
partition ij_l_odate49 values less than (to_date('1996-01-
01','YYYY-MM-DD'))
,
partition ij_l_odate50 values less than (to_date('1996-02-
01','YYYY-MM-DD'))
,
partition ij_l_odate51 values less than (to_date('1996-03-
01','YYYY-MM-DD'))
,
partition ij_l_odate52 values less than (to_date('1996-04-
01','YYYY-MM-DD'))
,
partition ij_l_odate53 values less than (to_date('1996-05-
01','YYYY-MM-DD'))
,
partition ij_l_odate54 values less than (to_date('1996-06-
01','YYYY-MM-DD'))
,
partition ij_l_odate55 values less than (to_date('1996-07-
01','YYYY-MM-DD'))
,
partition ij_l_odate56 values less than (to_date('1996-08-
01','YYYY-MM-DD'))
,
partition ij_l_odate57 values less than (to_date('1996-09-
01','YYYY-MM-DD'))
,
partition ij_l_odate58 values less than (to_date('1996-10-
01','YYYY-MM-DD'))
,
partition ij_l_odate59 values less than (to_date('1996-11-
01','YYYY-MM-DD'))
,
partition ij_l_odate60 values less than (to_date('1996-12-
01','YYYY-MM-DD'))
,
partition ij_l_odate61 values less than (to_date('1997-01-
01','YYYY-MM-DD'))
,
partition ij_l_odate62 values less than (to_date('1997-02-
01','YYYY-MM-DD'))
,
partition ij_l_odate63 values less than (to_date('1997-03-
01','YYYY-MM-DD'))
,
partition ij_l_odate64 values less than (to_date('1997-04-
01','YYYY-MM-DD'))
,
partition ij_l_odate65 values less than (to_date('1997-05-
01','YYYY-MM-DD'))
,
partition ij_l_odate66 values less than (to_date('1997-06-
01','YYYY-MM-DD'))
,
partition ij_l_odate67 values less than (to_date('1997-07-
01','YYYY-MM-DD'))
,
partition ij_l_odate68 values less than (to_date('1997-08-
01','YYYY-MM-DD'))
,
partition ij_l_odate69 values less than (to_date('1997-09-
01','YYYY-MM-DD'))
,
partition ij_l_odate70 values less than (to_date('1997-10-
01','YYYY-MM-DD'))
,
partition ij_l_odate71 values less than (to_date('1997-11-
01','YYYY-MM-DD'))
,
partition ij_l_odate72 values less than (to_date('1997-12-
01','YYYY-MM-DD'))
,
partition ij_l_odate73 values less than (to_date('1998-01-
01','YYYY-MM-DD'))
,
partition ij_l_odate74 values less than (to_date('1998-02-
01','YYYY-MM-DD'))
,
partition ij_l_odate75 values less than (to_date('1998-03-
01','YYYY-MM-DD'))
,
partition ij_l_odate76 values less than (to_date('1998-04-
01','YYYY-MM-DD'))
,
partition ij_l_odate77 values less than (to_date('1998-05-
01','YYYY-MM-DD'))
,
partition ij_l_odate78 values less than (to_date('1998-06-
01','YYYY-MM-DD'))
,
partition ij_l_odate79 values less than (to_date('1998-07-
01','YYYY-MM-DD'))
,
```

```
partition ij_l_odate80 values less than (to_date('1998-08-
01','YYYY-MM-DD'))
,
partition ij_l_odate81 values less than (to_date('1998-09-
01','YYYY-MM-DD'))
,
partition ij_l_odate82 values less than (to_date('1998-10-
01','YYYY-MM-DD'))
,
partition ij_l_odate83 values less than (to_date('1998-11-
01','YYYY-MM-DD'))
,
partition ij_l_odate84 values less than (MAXVALUE)
)
;
}
*sql
{
connect tpcd/tpcd
set timing on
set echo on

rem drop index i_l_pkey;
create index i_l_pkey
on lineitem
(l_partkey,l_shipmode,l_shipinstruct,l_quantity,l_extende
dprice,l_discount)
storage(freelists 16  initial 416320k next 416320k
pctincrease 0 minextents 1 maxextents unlimited)
nologging
pctfree 2
initrans 10
compute statistics
parallel 8;
}
*sql
{
connect tpcd/tpcd
set timing on
set echo on

rem 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
storage(freelists 16  initial 44464k next 44464k
pctincrease 0 minextents 1 maxextents unlimited)
nologging
initrans 10
compute statistics
parallel 8
local
;
}
*sql
{
connect tpcd/tpcd
set timing on
set echo on

rem 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
storage(freelists 16  initial 33368k next 33368k
pctincrease 0 minextents 1 maxextents unlimited)
nologging
initrans 10
compute statistics
parallel 8
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 ('IRAO')
,
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
set timing on
set echo on

rem drop index ij_l_okey;
create index ij_l_okey
on ma_qty (l_orderkey)
local
pctfree 2
storage(freelists 16  initial 50m next 50m pctincrease 0
minextents 1 maxextents unlimited)
nologging
initrans 10
compute statistics
parallel 8
;
}
*wait
*time{Index creation end}
*bgoff
%e-ixcre
%b-chob
*bgon=1
###############################################
##############################
# Change objects properties if needed
*time{Chob start}
*sql
{
connect tpcd/tpcd
set timing on
set echo on

alter table parts parallel 32;
}
*time{Chob end}
*bgoff
%e-chob
%b-anlyz
*bgon=1
###############################################
##############################
# Analyze Phase
*time{Analyze start}
*sql
{
connect tpcd/tpcd
set timing on
set echo on

execute dbms_stats.gather_schema_stats('tpcd' ,
estimate_percent => 1 , degree => 8 );

connect / as sysdba;
execute dbms_stats.gather_system_stats;
}
*wait
*time{Analyze end, i.e., Load End}
%e-anlyz
```

Bumpx.pl

```
#!/usr/bin/perl
#
# $Header: bumpx.pl 31-jan-00.16:09:54 mpoess Exp $
#
# bumpx.pl
#
#  Copyright (c) Oracle Corporation 1999, 2000. All
Rights Reserved.
#
#    NAME
#      bumpx.pl - <one-line expansion of the name>
#
#    DESCRIPTION
#      <short description of component this file
declares/defines>
#
#    NOTES
#      <other useful comments, qualifications, etc.>
#
#    MODIFIED   (MM/DD/YY)
#    lvaz        03/20/01 - added support for
tab_<name>_flatfile_area
#    mpoess      01/31/00 - fix bug with undo
#    mpoess      01/21/00 - fix bugs for partitioned
tablespaces
#    mpoess      01/20/00 - add OPS load one partiton from
one node support
#    mpoess      01/07/00 - add support for flatfile
distribution
```

```
# mpoess 01/06/00 - add controlfile generation performace improvement
# mpoess 12/16/99 - add partitioning support for control files
# mpoess 10/28/99 - adjust bumpx to fixed dbgen (84 partitioning)
# mpoess 08/17/99 - change column definition of base tables
# mpoess 08/13/99 - add support for reporting files
# mpoess 08/08/99 - change file atributes
# mpoess 07/12/99 - add setpath for dbgen
# mpoess 07/12/99 - disable -o option for dbgen
# mpoess 07/07/99 - fix dbgen phase
# mpoess 07/06/99 - change path to perl
# mpoess 07/02/99 - add environment variable support for conf file
# mpoess 06/28/99 - add pathname for param.txt file
# mpoess 06/28/99 - Copy from TPCD
# mpoess 06/28/99 - Creation
# MODIFIED (MM/DD/YY)
# mpoess 11/18/98 - add undo rollback segment support for OPS systems
# mpoess 11/18/98 - add control log etc. keywords in SQLLDR call
# mpoess 11/06/98 - fix bug in print_as_select
# mpoess 11/05/98 - change name of BUMPC_CTR variable
# mpoess 11/04/98 - delete default startup before index creation
# mpoess 11/04/98 - adopt Barry Perkins changes to dbgen
# mpoess 11/03/98 - change as select handling
# mpoess 11/02/98 - change background process handling for NT
# mpoess 10/26/98 - modifiy dapop so that multiple partitions can be lo
# mpoess 10/23/98 - take *waits out from flat file generation
# mpoess 10/20/98 - add sdgen, dbgen and dapop description
# mpoess 10/12/98 - fix bug in constraint creation
# mpoess 10/12/98 - add support for non partitioned tables
# mpoess 10/08/98 - add -including new values- clause in view log defin
# mpoess 09/27/98 - update links,param2html
# mpoess 09/25/98 - adding new features
# akarasik 07/29/98 -
# akarasik 07/29/98 - Checking in
# 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
#*********************************************
*******************************
#*********************************************
*******************************

$os = $ENV{'OS'};
if (($os cmp 'Windows_NT') != 0)
{                                       # os is UNIX
   $os = "unix";
   $nt = 0;
   $unix = 1;
}
else
{
   $os = "nt";
   $nt = 1;
   $unix = 0;
}

$| = 1;
$verbose = 0;
$allphases =
"dbcre,shutd,start,dbgen,sccre,scuto,scuvo,sctso,dapop,ixcre,anlyz,chob,expln,query,sdgen,plcre";
if (($os cmp "unix")==0)
{
   $defphases =
"dbcre,sctso,scuto,dbgen,dapop,anlyz,ixcre";
}
else
{
```

```
   $defphases =
"sdgen,shutd,start,dbgen,plcre,dbcre,sctso,scuto,dapop,sc
uvo,anlyz,ixcre,chob";
}
$allbmtypes = "tpcd,wisc";
$bmtype = "tpcd" if !defined $bmtype;
$pdfile =  "$ENV{'BUMPX_DIR'}/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/);
   $bmtype = shift(@ARGV) if ($arg =~ /-t/);
   $docreate = 1 if (($arg =~ /-c/) || ($arg =~ /-xc/));
   $doexecute = 1 if (($arg =~ /-x/) || ($arg =~ /-cx/));
   $phaselist = shift(@ARGV) if ($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/)
   {
         $infile = "$ENV{'BUMPX_DIR'}/bumpx.conf"
if !defined $infile;
         $infile = $infile;
         &readfile;
         &defaults;
         @keys = keys %params;
         while ($#keys >=0)
         {
            $key = pop(@keys);
            print $key, "=", $params{$key}, "\n";
         }
         exit(0);
   }
}

$docreate = 0 if !defined $docreate;
$doexecute = 0 if !defined $doexecute;
if (!($docreate || $doexecute))
{
   $error = "** Error: Must specify either -c or -x or
both\n";
   &usage;
}
$infile = "$ENV{'BUMPX_DIR'}/bumpx.conf" if
!defined $infile;
$outfile = "$ENV{'BUMPX_DIR'}/bumpx.dat" if
!defined $outfile;
if ($nt) {
   $listdir = $filedir."list/";
   if (!-e $listfile) {
         system ("mkdir $listdir");
   }
}
if (($os cmp "nt") == 0)
         {
      ## NT Port (Use tmpfile to buffer commands and
nruntpb to synchronize them)
            $tmpfile = "tmp.txt";
            $tmpfile = $filedir.$tmpfile;
            $nruntpb =  "nruntpb.exe";
      ## NT End
         }

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

if ($docreate)
{
   open (OUTFILE, ">$outfile") if $docreate;
   &readfile;
   &defaults;
   &docreate;
   close (OUTFILE);
}

## NT Port (Use tmpfile to buffer commands for nruntpb)
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)
      {
         &dump0($line) if $docreate;
         next WLOOP;
      }
      next WLOOP if $line =~ /^\s*\#/;
      next WLOOP if $line =~ /^\s*\n/;
      if ($contin)
      {
         if ($line =~ /(.*)\&\s*\n/) # still continuing
(changed \ to &)
         {
#             $pval = $pval . $1 . "\n";
            $pval = $pval . $1;
            next WLOOP;
         }
         $line =~ /(.*)\s*\n/; # reached the end
         $pval = $pval . $1;
         $pval =~ s/\?/$ENV{'ORACLE_HOME'}/g;
              &key_exists($pkey);
              if ($result!=1)
              {
                    print "Parameter $pkey does not
exist.\nBailing out!\n";
                    exit (0);
              }
              $params{$pkey} = $pval;
         $contin = 0;
         $pkey = "";
         $pval = "";
         next WLOOP;
      }
      else
      {
         if ($line =~ /\s*(\S+)\s*=\s*(.*)\&\s*\n/)
         {
            $pkey = $1;
#             $pval = $2 . "\n";
            $pval = $2;
            $contin = 1;
            next WLOOP;
         }
         if ($line =~ /\s*(\S+)\s*=\s*\n/)
         {
            undef($params{$1});
            next WLOOP;
         }
         if ($line !~
/\s*(\S+)\s*=\s*((\S+)|(\S+.*\S+))\s*\n/)
         {
            print "Bad line: $line";
            next WLOOP;
         }
         $tkey = $1;
         $tval = $2;
         $tval =~ s/\?/$ENV{'ORACLE_HOME'}/g;
              if ($tval =~ /\$/) {  # a $ sign at the beginning
of the contents
                                    # of the parameter value
results in a environment
                                    # variable lookup and substitution
                   $tenv = $tval;
                   $tval =~ s/\$//g;
                   if ($unix) {
                      $tenv =~ s/(.*)\/\$(.*)\/(.*)/\$$2/g;
                   }
                $tenvval = `echo $tenv`;
                   $tenv =~ s/\$//g;
                   chop($tenvval);
                   if ($tenvval=~/^\n/) {
                      print "bumpx error: Environment
variable $tval not defined!\nbailing out...\n";
                      exit(1);
                   }
                   $tval =~ s/$tenv/$tenvval/g;
              }

              &key_exists($tkey);
              if ($result!=1)
              {
                    print "Parameter $tkey does not
exists.\nBailing out!\n";
                    exit (0);
              }
```

```
            $params{$tkey} = $tval;
        }
    }
    close (INFILE);
}

sub docreate
{
    print "Creation pass begun." if $verbose;
    @phases_tmp = @phases; # because I will eat the
elements
    while ($phase=shift(@phases_tmp))
    {
        if ($allphases =~ /$phase/)
        {
                $nextphase = shift(@phases_tmp);
                unshift (@phases_tmp,$nextphase) if
($nextphase);
                print "\n  Creating phase \'$phase\'" if
$verbose;
            &dump0("%b-$phase\n");
            &prepmulti();
            &dump0("*bgon=$params{$phase.'_max_bg'}");
            eval (&$phase);
            &dump0("*wait");
            &dump0("*bgoff");
            &dump0("%e-$phase\n");
        }
        else
        {
            print "\n  Phase \'$phase\' not built in...assuming a
\*match block being used";
        }
    }
    print "\nCreation pass complete.\n" if $verbose;
}

sub doexecute
{
# First, do preprocessing stuff
    print "Execution pass begun." if $verbose;
    open (INFILE, $outfile);
  WLOOP1:
    while ($line = <INFILE>)
    {
        study $line;
        next WLOOP1 if $line =~ /^\s*\#/;
        next WLOOP1 if $line =~ /^\s*\n/;
        if ($line =~ /^%b-preproc/)
        {
            $insection = 1;
            next WLOOP1;
        }
        next WLOOP1 if ($insection != 1);
        if ($line =~ /^%e-preproc/)
        {
            $insection = 0;
            $commands{$shortcmd} = $longcmd if defined
$shortcmd;
            last WLOOP1;
        }
        if ($line =~ /^\*/)
        {
            $commands{$shortcmd} = $longcmd if defined
$shortcmd;
            $line =~ /^(\*.*\S+)\s*\n$/;
            $shortcmd = $1;
            $longcmd = "";
            next WLOOP1;
        }
        if ($line =~ /^\\/)
        {
            $line =~ /\\(.*\n)/;
            $longcmd = $longcmd . $1;
            next WLOOP1;
        }
        print "Illegal entry in preproc stage:\n   $line";
    }
    close (INFILE);

# Then, do all of the requested phases
    $execctr = 0;
    foreach $phase (@phases)
    {
            $phase_cmd_num = 0;
            print "\n  Executing phase \'$phase\'" if $verbose;
            $bg = 0;
        open (INFILE, $outfile);
      WLOOP2:
        while ($line = <INFILE>)
        {
            study $line;
            next WLOOP2 if $line =~ /^\s*\#/;
            next WLOOP2 if $line =~ /^\s*\n/;
            if ($line =~ /^\*ignon/)
            {
                $ignon = 1;
                next WLOOP2;
            }
            if ($line =~ /^\*ignoff/)
            {
                $ignon = 0;
                next WLOOP2;
            }
            next WLOOP2 if ($ignon == 1);
            if ($line =~ /^%b-$phase/)
            {
                $insection = 1;
                $execcmd = "";
                next WLOOP2;
            }
            next WLOOP2 if ($insection != 1);
            if ($line =~ /^%e-$phase/)
            {
                $insection = 0;
```

```
            &execute ($execcmd);
            last WLOOP2;
         }
         if ($line =~ /^\*(.*)/)
         {
            &execute ($execcmd);
            if (($1 =~ /bgo/) || ($1 =~ /wait/) || ($1 =~
/ignore/))
            {
               $execcmd = $line;
               next WLOOP2;
            }
            $line =~ /^(\*.*\S+)\s*\n$/;
            $execcmd = $commands{$1};
            next WLOOP2;
         }
         if ($line =~ /^\{(.*)\}/)
         {
            $insert = "";
            $insert = $1;
            $execcmd =~ s/\{\}/$insert/;
            next WLOOP2;
         }
         if ($line =~ /^\{(.*)$/)
         {
            $insubsection = 1;
            $insert = "";
            $insert = $1;
            next WLOOP2;
         }
         if ($line =~ /^(.*)\}/)
         {
            $insubsection = 0;
            $insert = $insert . $1;
## NT Port (Ignore '\n')
                  if (($os cmp "nt") == 0)
                  {
                     $insert =~ /(.*)\n$/s;
                     $insert = $1;
                  }
## NT End
            $execcmd =~ s/\{\}/$insert/;
            next WLOOP2;
         }
         $insert = $insert . $line if ($insubsection == 1);
      }
      close (INFILE);
   }
   print "\nExecution pass complete.\n" if $verbose;
}

sub execute
{
   $cmd = shift(@_);
   if ($cmd)
   {
      return if ($cmd =~ /^\*ignore/);
      if ($cmd =~ /^\*bgon=(.*)/)
      {
         $bgmax = $1;
         $bg = 1;
         $bgrun = 0;
         return;
      }
      if ($cmd =~ /^\*bgoff/)
      {
         $bg = 0;
         return;
      }

        if ($cmd =~ /^\*time=(.*)/)  ##NT only
      {
            print $1 . "\n";
            print localtime(time) . "\n";
         return;
      }
      if ($cmd =~ /^copy (.*)/) ## NT only
      {
            system($cmd);
         ## Quit if failed
         if ($?) {
                  print "system copy command
failed:\n$cmd\nreason: $? ($!)\n";
               exit(-1);
            }
         return;
      }
        if ($cmd =~ /^del (.*)/) ## NT only
      {
            system($cmd);
         ## Quit if failed
         if ($?) {
                  print "system del command
failed:\n$cmd\nreason: $? ($!)\n";
               exit(-1);
            }
         return;
      }

         if ($cmd =~ /^\*wait/) ## This deals with main
differences between NT and UNIX
      {
            if (($os cmp "unix") == 0)
            {
                  while ($fpid = shift(@wpids))
                  {
                     waitpid($fpid, 0);
                  }
            }
            else
            {     ## NT Port (Start background tasks if any.
nruntpb will wait until all tasks are done)
                  if ($bgrun >= 1)
                  {
                     close(TMPFILE);
```

```
                        system("cat $tmpfile >>
$listdir$phase.lst");
                        system("vi $tmpfile") if $debug;
                        system("$nruntpb -p < $tmpfile") if
!$debug;
                        if ($?)
                        {
                                print "system command
failed:\n$nruntpb < $tmpfile\n";
                                print "reason: $? ($!)\n";
                                print "Please check the
contents in the input file.\n";
                                exit(-1);
                        }
                        open(TMPFILE, ">$tmpfile");
                    }
                }
            $bgrun = 0;
            return;
        }

        if ($cmd =~ /(s|g)etenv/)
        {
                @lines = split(/\n/, $cmd);
            $cmd = "";
            foreach $line (@lines)
            {
                while (1)
                {
                    last if ($line !~ /getenv/);
                    $line =~ /(.*)\*getenv\(([^(\)|\*)]*)\)(.*)/;
                    $line = $1 . $ENV{$2} . $3;
                }
                if ($line =~ /jojo/) #we do not want to use this
for now
                {
                    $line =~ /setenv\s+(\S+)\s+(\S+)/;
                    $ENV{$1} = $2;
                }
                else
                {
                    $cmd = $cmd . $line. "\n";
                }
            }
        }
        return if ($cmd !~ /\S+/); # return if nothing left to
execute

        $execctr++;
        $ENV{'BUMPX_CTR'} = $$.'-'.$execctr;

            if (($os cmp "unix") == 0)
            {
                if ($bg == 1)
                {
                        print "." if $verbose;
                        $fpid = fork;
                        if ($fpid == 0)
                        {
                            exec ($cmd);
                            print "exec\'d command
failed:\n$cmd\nreason: $!\n";
                            exit(-1);
                        }
                        unshift (@wpids, $fpid);
                        $bgrun = $bgrun + 1;
                        &execute ("*wait") if (($bgrun >=
$bgmax) && ($bgmax >= 0));
                    }
                    else
                    {
                        system ($cmd);
                        print "system\'d command
failed:\n$cmd\nreason: $? ($!)\n" if $?;
                    }
                }
                else ## NT support
                {
            ## NT Port (Submit background tasks if there are
bgmax of them, otherwise write to tmpfile)
                    if ($bg == 1)
                    {
                        print "." if $verbose;
                        if ($bgrun < $bgmax)
                        {
                            $cmd =~
s/phase\#.lst/$listdir$phase\_$phase_cmd_num.lst/;
                            ++$phase_cmd_num;
                            print TMPFILE $cmd;
                            $bgrun = $bgrun + 1;
                        }
                        else
                        {
                            close(TMPFILE);
                            system("cat $tmpfile >>
$listdir$phase.lst");
                            system("$nruntpb -p < $tmpfile");
                            if ($?) {
                                print "system command
failed:\n$nruntpb < $tmpfile\nreason: $? ($!)\n";
                                print "Please check the
contents in the input file.\n";
                                exit(-1);
                            }
                            open(TMPFILE, ">$tmpfile");
                            $cmd =~
s/phase\#.lst/$listdir$phase\_$phase_cmd_num.lst/;
                            ++$phase_cmd_num;
                            print TMPFILE $cmd;
                            $bgrun = 1;
                        }
                    }
                    else
                    {
                        $cmd =~
s/phase\#.lst/$listdir$phase\_$phase_cmd_num.lst/;
```

```
                    ++$phase_cmd_num;
                    print TMPFILE $cmd;
                    close(TMPFILE);
                    system("cat $tmpfile >>
$listdir$phase.lst");
                    system ("sh $tmpfile");
                    if ($?) {
                       print "system\'d command failed:\nsh
$tmpfile\nreason: $? ($!)\n";
                       print "Please check the contents in the
shell script.\n";
                       exit(-1);
                    }
                    open(TMPFILE, ">$tmpfile");
                }
            }  ## NT support End
    }
}

sub usage
{
    print "Usage:\n";
    print " $0 [-c] [-x] [-i infile] [-o outfile] [-p phaselist] [-t
type]\n";
    print "   -c : create intermediary file (needed for
execution)\n";
    print "   -x : execute intermediary file\n";
    print "   -i : configuration file to be used\n";
    print "        defaults to bumpx.conf in \$BUMPX_DIR
or \$CWD\n";
    print "   -o : intermediary file to be created and/or
used\n";
    print "        defaults to bumpx.dat in \$BUMPX_DIR or
\$CWD\n";
    print "   -p : list of phases to create/execute\n";
    print "        phaselist is a comma separated list of phases
in order\n";
    print "        possible phases are:\n";
    print "          sdgen = seed file generation\n";
    print "          dbgen = data flat file generation\n";
    print "          plcre = NT raw partition and links
creation\n";
    print "          dbcre = database creation\n";
    print "          shutd = shutdown database (on all
instances)\n";
    print "          start = startup database (on all
instances)\n";
    print "          sccre = schema creation\n";
    print "          sctso = schema creation (tablespaces
only)\n";
    print "          scuto = schema creation (user and tables
only)\n";
    print "          scuvo = schema creation (views only)\n";
    print "          dapop = data population\n";
    print "          ixcre = index creation (including
constraints)\n";
    print "          anlyz = analyze objects\n";
    print "          chob  = change parameters of objects\n";
    print "          expln = create explain plans\n";
    print "          query = run and time queries\n";
    print "        defaults to $defphases\n";
    print "   -t : type of benchmark\n";
    print "        enables benchmark-specific defaults\n";
    print "        current possibilties are: $allbmtypes\n";
    print "        defaults to tpcd\n";
    print "   -s : run silent (no parameter checking is
done)\n";
    print "\n";
    print "Examples:\n";
    print " $0 -c -o file.out\n";
    print "   This will create an intermediary file named
file.out for the default\n";
    print "   phases using bumpx.conf which can be
executed in the future.\n";
    print " $0 -x -p dapop\n";
    print "   Executes data population phase of intermediary
file bumpx.dat.\n";
    print " $0 -cx -p dbcre,dapop\n";
    print "   This will create an intermediary file,
bumpx.dat, using configuration\n";
    print "   file, bumpx.conf, for both the database creation
phase and the data\n";
    print "   population phase, and then execute that file.\n";
    print "\n";
    print "$error\n";
    exit(-1);
}

#***********************************************
*******************************
#***********************************************
*******************************
# All Known Phases
#***********************************************
*******************************
#***********************************************
*******************************

sub dbcre
{

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

# Shut down anything that may have been running
    &shutdb();

# 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= $params{'startupfile_dbcre'}
nomount;\n";
   $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_rs1 ";
   $cmd .= "  storage $params{'ts_undo_rs_storage'}" if
defined $params{'ts_undo_rs_storage'};
   $cmd .= ";\n";
   $cmd .= "\n";
   $cmd .= "alter rollback segment t_rs1 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");
   startdb("dbcre");
   &dump0("*wait");
   @ops_nodes =  split(/,/,$params{'ops_nodes'});
   if (defined $params{'ops_nodes'}) {
          foreach $ops_node (@ops_nodes)
          {
             if ($ops_node =~ /undo/)
             {
                    $ts_undo = 'undo';
             }
             else {
                    $ts_undo = 'undo_'.$ops_node;
             }
             if ($params{'skip_ts'} !~ /$ts_undo/)
             {
                    &add_ts_rollb ($ts_undo);
             }
          }
   } else {
          $ts_undo = 'undo';
   }


   # 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'}; $j++)
      {
         (($j+1) == $params{'ts_log_#files_pt'}) ?
($addcomma = ";") : ($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 .= "connect system/manager\n";
   $cmd .= '@' . "$params{'dd_sql_area'}" .
"utlxplan.sql;\n";
   $cmd .= '@' . "$params{'dd_sqlplus_area'}" .
"pupbld.sql;\n";
   &dump1("*sql");
   &dump0("*wait");
   &time0("Done database creation");

# prepare for multi-user
#^^sd
   $cmd .= "shutdown;\n";
   &dump1("*sql");
   &dump0("*wait");
   &startdb("dbcre");
}                                       # 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($nextphase);
   &time0("Done database startup - all instance");
}

sub sdgen
{

&dump0("#####################################
########################################");
   &dump0("# Data (Seed File) Generation Phase");
   &time0("Begin creating seed files");
   @load_tablelist = split(/,/, $params{'load_tables'});
   foreach $table (@load_tablelist)
   {
          $bgopt = "";
          $dbgprc  =
$params{'load_dbgen_'.$table.'_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");

# change DSS_PATH
   $cmd = "setenv DSS_PATH
$params{'load_flatfile_area'}";
   &dump1("*sh");

   $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)
   {
      $curts = $params{'tab_'.$table.'_ts'};
      $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;

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

#lvaz : 03/20/01
```

```
# If the parameter tab_<name>_flatfile_area is present,
then check if the
# number of entries in it is equal to the number of loaders

      $dplist_ff_num=0;
      @dplist_ff  = split(/,/,
$params{'tab_'.$table.'_flatfile_area'});
      $dplist_ff_num = scalar@dplist_ff if defined
$params{'tab_'.$table.'_flatfile_area'};
      if ( defined $params{'tab_'.$table.'_flatfile_area'} &&
$dbgpar != $dplist_ff_num)
      {
        print(" Bailing out...insufficient number of flatfile
locations \n");
        exit(0);
      }

#lvaz : 03/20/01

# 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 ($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 .=
$params{'load_dbgen_'.$table.'_option_C'};
           # if using pipes, create the pipes

           $nperset = 1;

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

           for ($i=0;$i<$dbgpar;$i++)
           {
              # create script to generate flat files for each
dbgpar
              $cmd = sprintf("%s -S %d", $dbgopt, ($i +
1));
              $fpre =
$params{'load_flatfile_area'}.$params{'load_dbgen_'.$tab
le.'_output_prefix'};
#lvaz : 03/20/01
# extract the flatfile location for each dbgen

         if ( defined $params{'tab_'.$table.'_flatfile_area'} )
         {
           $fpre_temp=shift@dplist_ff;

$fpre=$fpre_temp.$params{'load_dbgen_'.$table.'_output
_prefix'};
         }

#lvaz : 03/20/01
              if ($dgp)
              {
                    # create script to generate partitioned
flat files

                    # if the output prefix name is specified
with #
                    # divide output file names into
$dp/#numpart sets
```

```
                    if ($fpre =~ /\#/)
                    {
                       $snum = 1;
                       $snum = (int($i/$nperset) + 1) if ($dp
> 1);
                       $fpre =~ s/\#/$snum/g;
                    }
                    # I commented this out to enable
support for fixed dbgen (fixed to 84 partitions)
                    $cmd .= " -p -I ";
                    $cmd .=
$params{'load_dbgen_'.$table.'_input_params'};
                    $cmd .= sprintf(" -o %s_%d ", $fpre,($i
+ 1));
                    # the folowing is the new stuff
                    #$cmd = "-D ".$cmd;

             } else {
                    # $cmd .= sprintf(" -o $fpre"); this is
only temporary
             }
             $cmd = $verbose ? "-v -f " . $cmd : "-f " .
$cmd;
#            $cmd .= " & ";
             $cmd .= "\n";
             &dump1("*dbgen");
          }
          &dump0("*wait");
          &time0("Done creating flat files for table
$table");
   }
   &dump0("*wait");
   &time0("Done creating flat files");
} # end of dbgen


sub dbgen_old
{

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

   $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)
   {
          $dbgprc =
$params{'load_dbgen_'.$table.'_option_C'};
      $curts = $params{'tab_'.$table.'_ts'};
      $dp = $params{'tab_'.$table.'_load_degpar'};
      $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 =
$params{'load_dbgen_'.$table.'_option_C'};

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

          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'};
        }

        # 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
$params{'load_dbgen_flatfile_backup'}) ?
           $params{'load_dbgen_flatfile_backup'} :
           0;
        $bpre = (defined
$params{'load_dbgen_backup_'.$table.'_prefix'}) ?

$params{'load_dbgen_backup_'.$table.'_prefix'} :
           $params{'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)
           {
                $nset = ($params{'tab_'.$table.'_#part'}
> $dp) ? 1 : int($dp / $params{'tab_'.$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
$params{'load_dbgen_'.$table.'_output_prefix'}) ?
           $params{'load_dbgen_'.$table.'_output_prefix'} :
                        $params{'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          #
        &time0("Done creating flat files for table
$table");
   }
   &dump0("*wait");
   &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 $multi;  # for possible locality on SP2
   ($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)
   {
      $curts = $params{'tab_'.$table.'_ts'};
      $dp = $params{'tab_'.$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

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


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


         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";
         }

      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 dbgpar

            $nperset = 1;

            if ($dgp && $dp > 1) {
                $nset = int($dp /
$params{'tab_'.$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
$params{'load_dbgen_'. $table.'_output_prefix'}) ?
$params{'load_dbgen_'.$table.'_output_prefix'} :
$params{'load_dbgen_def_output_prefix'};
                    if ($fpre =~ /\#/) {
                        $snum = 1;
                        $snum = (int($i/$nperset) + 1)
if ($dp > 1);
                        $fpre =~ s/\#/$snum/g;
                    }
                    $cmd .= sprintf(" -o %s_%d ",
$fpre,($i + 1));
                }

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

                $cmd .= "\n";
                &dump1("*dbgen");
            }

            &dump0("*wait");

        # create script to concatenate partitioned flat files
for each dbgpar 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<$dbgpar;$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 =~ /\#/) {
                        $snum = 1;
                        $snum = (int($i/$nperset) + 1)
if ($dp > 1);
                        $fpre =~ s/\#/$snum/g;
                    }
                    $pnum = ($i % $nptn) + 1;
                    $cmd = sprintf("copy %s_*.%d
%s.%d\n", $fpre, $pnum, $fpre, $pnum);
                    &dump1("*sh");
                    &dump0("*wait");
                    $cmd = sprintf("del %s_*.%d\n",
$fpre, $pnum);
                    &dump1("*sh");
                    &dump0("*wait");
                }
            }
            else
        {
            $cmd = "-f " . $dbgopt;
            $cmd .= "\n";
            &dump1("*dbgen");
        }
            &dump0("*wait");
            &time0("Done creating flat files for table
$table");
    }
    &dump0("*wait");
    &time0("Done creating flat files");
} # dbgen_nt


sub dbgen_unix
{
```

```
&dump0("########################################
########################################");
   &dump0("# 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)
   {
      $curts = $params{'tab_'.$table.'_ts'};
      $dp = $params{'tab_'.$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

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

      $params{'tab_'.$table.'_load_ctlf'} = $table.'.ctl' if
!defined $params{'tab_'.$table.'_load_ctlf'};
      @ctlf = split(/,/, $params{'tab_'.$table.'_load_ctlf'});
      $err = "ctl" 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;
      @datf = split(/,/,
$params{'tab_'.$table.'_load_datf'});
      $err = "dat" if ((@datf > 1) && (@datf != $dp));
      $dafexist = 1 if @datf > 0;
      @disf = split(/,/, $params{'tab_'.$table.'_load_disf'});
      $err = "dis" if ((@disf > 1) && (@disf != $dp));
      $difexist = 1 if @diff > 0;
           @filf = ();
#       $params{'tab_'.$table.'_load_filf'} =
$params{$curts.'_datafiles'} if
($params{'tab_'.$table.'_load_filf'} =~ /alltsdatafiles/);
      if ($params{'tab_'.$table.'_load_filf'} =~
/alltsdatafiles/)
      {
       $params{'tab_'.$table.'_load_filf'} =
$params{$curts.'_datafiles'}
      }
      @filf = split(/,/, $params{'tab_'.$table.'_load_filf'}) if
defined $params{'tab_'.$table.'_load_filf'};
           # relax the requirement a little bit by allowing
round robining
           $numfil = @filf;
#          if (($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);
           }
      $err = "fil" if ((@filf > 1) && ($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;
      }
      $usels = 1 if (($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[$i] = @logf[0];
           @lf[$i] =~ s/\#/$iplus/g;
        }
        if (@badf > 1)
        {
           @bf[$i] = @badf[$i];
        }
        else
        {
           @bf[$i] = @badf[0];
           @bf[$i] =~ s/\#/$iplus/g;
        }
        if (@datf > 1)
        {
           @daf[$i] = @datf[$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
                # sqlldr.

                if (($params{'tab_'.$table.'_#part'} > 0)
&&
                   ($dp >
$params{'tab_'.$table.'_#part'})) {
                   if ($dp %
$params{'tab_'.$table.'_#part'} == 0) {
                      if (@datf[0] =~ /\#.*\#/) {
                         # set number

$snum=int($i/$params{'tab_'.$table.'_#part'}) + 1;
                         # partiton number

$pnum=$i%$params{'tab_'.$table.'_#part'} + 1;
                         @daf[$i] = @datf[0];
                         @daf[$i] =~ s/\#/$snum/;
                         @daf[$i] =~ s/\#/$pnum/;
                      } else {
                         @daf[$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 $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)
        {
           @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'};
              $filnum = $i % $numfil + 1;
              @ff[$i] =~ s/\#/$filnum/g;
           } else {
              @ff[$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'};
     $pname = $params{'tab_'.$table.'_partnames'};
     @pname = split(/,/, $pname);
# create the controlfiles (fixed-length fields or delimited
records)
#       if ($params{'skip_mk_ldctlfs'} !~ /$table/)
#        {
```

```
        $npart = (defined $params{'tab_'.$table.'_#part'}) ?
$params{'tab_'.$table.'_#part'} : 1;
        for ($ictl=1; $ictl<=$npart; $ictl++) {
          open (CTLFILE, ">$control");
          print CTLFILE "---\n";
          print CTLFILE "---  $table.ctl for delimited
records\n" if ($ud == 1);
          print CTLFILE "---  $table.ctl for fixed-length
fields\n" if ($ud == 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
($params{'load_unrecoverable'} =~ /[tT][rR][uU][eE]/);
          print CTLFILE "load\n";
          print CTLFILE "-- This is where INFILE should
go.\n";
          if ((!($up)) && ($dbgpar)) {
            for ($jpart=1;$jpart<=$numchild;$jpart++) {
              print CTLFILE "INFILE
'".$ff_path.$jpart.".".$ictl."'\n";
            }
          }
          print CTLFILE "$params{'load_insert_type'}\n";
          print CTLFILE "into table $table\n";
          if ($dgp) {
            print CTLFILE "partition ($pname[$ictl-1])\n";
          }
          print CTLFILE "$params{'load_insert_type'}\n";
          print CTLFILE "fields terminated by
$params{'load_field_terminator'}\n" if ($ud == 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 ($ud == 0);
            $ctlline = sprintf ("  %-20s %s %s$addcomma",
$col, $pos, $params{'tab_'.$table.'_'.$col.'_loadcolx'});
            print CTLFILE "$ctlline\n";
          }
          print CTLFILE ")\n";
          close (CTLFILE);
          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++) {
                $cmd = sprintf("rm -f %s%s\nmknod %s%s
p\n",$params{'load_flatfile_area'},@daf[$i],$params{'loa
d_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'});
                  $nperset = int($dbgpar / $nset);
            }

            for ($i=0;$i<$dbgpar;$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%s",
$params{'load_controlfile_area'}, @cf[$i]);
         $control = sprintf (" control=/host%s%s",
$params{'load_controlfile_area'}, @cf[$i]) if
($params{'special_machine'} eq 'ncube');
         $log = sprintf (" log=%s%s",
$params{'load_otherfile_area'}, @lf[$i]) if $lfexist;
         $bad = sprintf (" bad=%s%s",
$params{'load_otherfile_area'}, @bf[$i]) if $bfexist;
```

```
        $data = sprintf (" data=%s%s",
$params{'load_flatfile_area'}, @daf[$i]) if $dafexist;
        $discard = sprintf (" discard=%s%s",
$params{'load_otherfile_area'}, @dif[$i]) if $difexist;
        $file = sprintf (" file=%s%s",
$params{$curts.'_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=99999999";
            }

        $cmd = sprintf
("%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s\n",$up
wd,$control,$log,$bad,$data,$discard,$dismax,$skip,$loa
d,$errors,$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 = $params{'plcre_setlinks_input_file'};
   open (LNKSFILE, ">$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 (LNKSFILE);
   &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 sccre
{

&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";
      $cmd .= "@?/rdbms/admin/utlxplan;\n" if (($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 ($phaselist !~ /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 ($phaselist !~ /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");
      }
          $cmd .= "connect tpcd/tpcd\n";
          $cmd .= '@' . "$params{'dd_sql_area'}" .
"utlxplan.sql;\n";
          &dump1("*sql");
          &dump0("*wait");
          &time0("Done creating user and tables (no
datafiles)");
   }
}


sub scuvo
{
   if ($phaselist =~ /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)
             {
                    $ob_type = 'viewlog';
                    $object = $viewlog;
                    if (!(defined
$params{'created_'.$object}) )
                    {
                       $params{'created_'.$object} =
$object;
                       &create_object;
                    }
             }
             $ob_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 dsffs
{

&dump0("#####################################
#########################################");
   &dump0("# Flatfile Distribution Phase");
   &time0("Begin data population");

   # &recycle_db() if ($params{'startup_db_dapop'});

   $cur_inst = 1 if $multi;  # for possible locality on SP2
   ($params{'load_type'} =~ /delim/) ? ($ud = 1) : ($ud =
0);

}

sub dapop # Meikel's version
{

&dump0("#####################################
#########################################");
   &dump0("# Database Population Phase");
   &time0("Begin data population");

   # &recycle_db() if ($params{'startup_db_dapop'});


   # find out how many nodes we have
   if (!(defined(@ops_nodes))) {
          $nnodes=1
   } else {
          $nnodes= ($#ops_nodes + 1);
   }

   $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)
   {
          &time0("Begin $table load");
      $curts = $params{'tab_'.$table.'_ts'};

# see if we are using pipes

          $up = $params{'load_use_pipes'} =~ /$table/ ? 1
: 0;

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

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

      $upwd = "$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'};

	$ff_path =
$params{'load_flatfile_area'}.$params{'load_dbgen_'.$tab
le.'_output_prefix'};
		$pname = $params{'tab_'.$table.'_partnames'};
		$pname =~ s/\#//g;
		print $pname."--".@pname;
		@pname = split(/\,/,$pname);
		$npart = ((defined
$params{'tab_'.$table.'_#part'}) && $dgp) ?
		   $params{'tab_'.$table.'_#part'} : 1;
		$ctlarea = $params{'load_controlfile_area'};
		$logarea = $params{'load_otherfile_area'};

		if ($dbgpar*$npart <= $dp)
		{
		   $dpmax = $dbgpar*$npart;
		   $numchild = 1;
		   $remchild = 0;
		}
		else
		{
		   $dpmax = $dp;
		   $numchild = int(($dbgpar*$npart)/$dp); #
numchild is the number of infiles of one loader
		   $remchild = ($dbgpar*$npart) - ($numchild *
$dp);
		   #print "dp numchild remchild ".$dp."
".$numchild." ".$remchild."\n";
		   $loaders_p_part=$dp/$npart;
		   #print "dp npart loaders per partition
".$loaders_p_part." ".$dp." ".$npart."\n";
		   $ninfiles=int($dbgpar/$loaders_p_part);
		   $ninfiles_last=$ninfiles+$dbgpar-($ninfiles *
$loaders_p_part);
		   #print "infiles lastloader ".$ninfiles."
".$ninfiles_last."\n";
		}
		# if we load on an ops we want to load each
partition only on one node

		$npartitions_per_node = int($npart / $nnodes); #
#partition/#nodes
		$rempartitions = $npart - ($npartitions_per_node
* $nnodes); #reminder of the partitions
		print "nnodes $nnodes npart $npart table $table
npartitions_per_node $npartitions_per_node rempartitions
$rempartitions\n";

		$node = 0;
	    $idbgpar = 0;
		$ipart=1;
		$iipart=1;
		if ($dp > $npart) {
		   print "$dp $npart\n";
		   $sqlldr_per_partition = int ($dp / $npart); #
load_degpar / #partitions
		   print "** dp $dp npart $npart
sqlldr_per_partition $sqlldr_per_partition\n";
		   $infiles_per_sqlldr = int ($dbgpar /
$sqlldr_per_partition ); #_C / sqlldr_per_partition
		   $sqlldr_per_partition_rest = $dbgpar -
$sqlldr_per_partition * $infiles_per_sqlldr;
		   print "** dbgpar $dbgpar sqlldr_per_partition
$sqlldr_per_partition\n";
		   print "** sqlldr_per_partition
$sqlldr_per_partition sqlldr_per_partition_rest
$sqlldr_per_partition_rest infiles_per_sqlldr
$infiles_per_sqlldr \n";
		}
		else{
		  $sqlldr_per_partition = 1;
		  $infiles_per_sqlldr = int ($dbgpar /
$sqlldr_per_partition ); #_C / sqlldr_per_partition
	    }

		$partcount = 1;
		if (defined(@ops_nodes)){
		   &sqlldr_ctl_ops; # this is a special controlfile
generation for OPS
		}
		else {
		for ($idp=1; $idp<=$dpmax; $idp++)
		{
		   if ($partcount > $sqlldr_per_partition) {
			$partcount = 1;
		   }
		   if (defined(@ops_nodes)){
			$node = $node + 1;
			if ($node > $#ops_nodes+1) {
			   $node = 1;
			}
		   }
		   else {
			$node = "";
		   }
```

```
        #$idbgpar = 0;
# create all of the "executable" load sections
        #$ipart= int(((($idp-
1)*$numchild)/$dbgpar)+1; # this is the first partition of
the new series
        &advmulti();
        $ctlfile = $params{'tab_'.$table.'_load_ctlf'};
        $ctlfile =~ s/\#/$idp/g;
        $control = $ctlarea.$ctlfile;
        if ($dgp) # load this table in partition mode
        {
                #$control = sprintf ("
%s%s_%d_%d.ctl", $ctlarea, $table, $idp, $ipart);
                $log = sprintf (" %s%s_%d_%d.log",
$logarea, $table, $idp, $ipart);
        }
        else
        {
                if ($dbgpar == 1) #dbgpar is option _C
for this table
                {
                   #$control = sprintf (" %s%s.ctl",
$ctlarea, $table);
                   $log = sprintf (" %s%s.log",
$logarea, $table);
                }
                else
                {
                   #$control = sprintf (" %s%s_%d.ctl",
$ctlarea, $table, $idp);
                   $log = sprintf (" %s%s_%d.log",
$logarea, $table, $idp);
                }
        }
        $cmd = sprintf ("%s control=%s%s%s
log=%s%s%s%s%s%s%s\n",

$upwd,$control,$skip,$load,$log,$errors,$rows,

$bsize,$silent,$dirbool,$parbool);
        &dump1("*load$node");
        &dump0("*wait") if ((!$parbool) &&
(!defined $params{'load_no_waits'}));

# create the controlfiles (fixed-length fields or delimited
records)
        if ($params{'skip_mk_ldctlfs'} !~ /$table/)
        {
                &load_ctl_head;
                $infiles = 0;
                $badfiles = 0;
                $discardfiles = 0;
                $intopartitions = "BEGINNING"; # this
will collect the partition numbers that will be served.
                if ($dp < $npart) {
                   if ($idp <= $remchild) {
                        $nchild = $numchild+1;
                   }else{
                        $nchild = $numchild;
                   }
                }else{
                   if ($partcount <=
$sqlldr_per_partition_rest) {
                        $nchild =
$infiles_per_sqlldr+1;
                   }else{
                        $nchild = $infiles_per_sqlldr;
                   }
                }
                for ($jpart=1; $jpart<=$nchild;
$jpart++)
                {
                   #print "ipart $ipart\n";
                   $idbgpar = 0 if $idbgpar >= $dbgpar;
                   #$ipart = int(((($idp-
1)*$nchild+$jpart)/$dbgpar);
                   #$ipart = $ipart+1 if ((((($idp-
1)*$nchild+$jpart)-$ipart*$dbgpar)>0);
                   if ($dgp)
                   {
                        $pn=@ipart[$node];
                        $partitionname = $pname.$pn;
                        if (!($intopartitions =~
/$partitionname/)) {
                           if ($intopartitions =~
/BEGINNING/)
                           {
                                $intopartitions =
$partitionname;
                           }
                           else {
                                $intopartitions .=
",".$partitionname;}
                        }
                        @infile[$infiles++] = """ .
$ff_path . "_" . ++$idbgpar .
                           "." . $ipart . """;
                        @badfile[$badfiles++] = """ .
$ff_path . "_" . $idbgpar .
                           "_" . $ipart . ".bad'";
                        @discardfile[$discardfiles++]
= """ . $ff_path . "_" . $idbgpar .
                           "_" . $ipart . ".dsc'";
                        #if ($remchild-- > 0)
                        #{
                        #   @infile[$infiles++] = """ .
$ff_path . "_" . ++$idbgpar .
                        #           "." . $ipart . """;
                        #   @badfile[$badfiles++] =
""" . $ff_path . "_" . $idbgpar .
                        #           "_" . $ipart . ".bad'";
                        #
@discardfile[$discardfiles++] = """ . $ff_path . "_" .
$idbgpar .
                        #           "_" . $ipart . ".dsc'";
                        #}
```

```
                        }
                        elsif ($dbgpar == 1)
                        {
                                @infile[$infiles++] = """ .
$ff_path . ".tbl"";
                                @badfile[$badfiles++] = """ .
$ff_path . ".bad"";
                                @discardfile[$discardfiles++]
= """ . $ff_path . ".dsc"";
                        }
                        else
                        {
                                @infile[$infiles++] = """ .
$ff_path . ".tbl." . ++$idbgpar . """;
                                @badfile[$badfiles++] = """ .
$ff_path . "_" . $idbgpar . ".bad"";
                                @discardfile[$discardfiles++]
= """ . $ff_path . "_" . $idbgpar .
                                    ".dsc"";
                                if ($remchild-- > 0)
                                {
                                   @infile[$infiles++] = """ .
$ff_path . ".tbl." . ++$idbgpar . """;
                                   @badfile[$badfiles++] = """
. $ff_path . "_" . $idbgpar . ".bad"";

@discardfile[$discardfiles++] = """ . $ff_path . "_" .
$idbgpar .
                                          ".dsc"";
                                }
                        }
                        #print "vorher jpart $jpart ipart $ipart
iipart $iipart dbgpar $dbgpar\n";
                        if ($iipart >= $dbgpar){
                                $iipart=0;
                                $ipart=$ipart+1;
                        }
                        $iipart=$iipart+1;
                        #print "nacher jpart $jpart ipart $ipart
iipart $iipart dbgpar $dbgpar\n";
                    }
                    --$infiles;
                    if ($up)
                    {
                       print CTLFILE "INFILE ";
                       for ($fpart=0; $fpart<=$infiles;
$fpart++)
                        {
                                print CTLFILE @infile[$fpart]
. ", " if $fpart < $infiles;
                                print CTLFILE @infile[$fpart]
. " " if $fpart == $infiles;
                        }
                        print CTLFILE "BADFILE " .
@badfile[0] . " ";
                        print CTLFILE "DISCARDFILE " .
@discardfile[0] . "\n";
                    }
                    else
                    {
                       for ($fpart=0; $fpart<=$infiles;
$fpart++)
                       {
                                print CTLFILE "INFILE " .
@infile[$fpart] . " ";
                                print CTLFILE "BADFILE " .
@badfile[$fpart] . " ";
                                print CTLFILE
"DISCARDFILE " . @discardfile[$fpart] . "\n";
                       }
                    }
                    print CTLFILE
"$params{'load_insert_type'}\n";
                    print CTLFILE "into table $table\n";
                    if ($dgp)
                    {
                       #print CTLFILE "partition
($pname$ipart)\n" if !($params{'tab_'.$table.'_parttype'}
=~ /hash/);
                       print CTLFILE "partition
($intopartitions)\n" if !($params{'tab_'.$table.'_parttype'}
=~ /hash/);
                    }
                    &load_ctl_tail;
                }
                $partcount = $partcount + 1;
            }
            &dump0("*wait") if !defined
$params{'load_no_waits'} && $dgp;
            &time0("End of load for Partition: $ipart for
Table: $table") if ($dgp);

        &dump0("*wait") if !defined
$params{'load_no_waits'};
    }   &time0("End $table load");
    }
    &dump0("*wait");
    &time0("Done data population");
} # end of dapop


sub sqlldr_ctl_ops
{
    $controlfiles_per_node = $dpmax / $nnodes;
    print "controlfiles_per_node $controlfiles_per_node
nnodes $nnodes\n";
    for ($i=1; $i<=$nnodes; $i++){
            @ipart[$i] = (($i-1)*$npartitions_per_node)+1;
            @iipart[$i] = 1;
            print "Offset for node $i is: @ipart[$i]\n";
    }
    for ($idpp=1; $idpp<=$controlfiles_per_node;
$idpp++) {
    for ($node=1; $node<=$nnodes; $node++)
    {
```

```
        print "idpp $idpp node $node  infiles_per_sqlldr
$infiles_per_sqlldr\n";
        $idp = (($node-1)*$controlfiles_per_node) +
$idpp;
        if ($partcount > $sqlldr_per_partition) {
           $partcount = 1;
        }
        #if (defined(@ops_nodes)){
        #    $node = $node + 1;
        #    if ($node > $#ops_nodes+1) {
        #            $node = 1;
        #    }
        #}
        #else {
        #    $node = "";
        #}
        #$idbgpar = 0;
# create all of the "executable" load sections
        #@ipart[$node]= int((($idp-
1)*$numchild)/$dbgpar)+1; # this is the first partition of
the new series
        &advmulti();
        $ctlfile = $params{'tab_'.$table.'_load_ctlf'};
        $ctlfile =~ s/\#/$idp/g;
        $control = $ctlarea.$ctlfile;
        if ($dgp) # load this table in partition mode
        {
           #$control = sprintf (" %s%s_%d_%d.ctl",
$ctlarea, $table, $idp, @ipart[$node]);
           $log = sprintf (" %s%s_%d_%d.log",
$logarea, $table, $idp, @ipart[$node]);
        }
        else
        {
           if ($dbgpar == 1) #dbgpar is option _C for this
table
           {
                #$control = sprintf (" %s%s.ctl",
$ctlarea, $table);
                $log = sprintf (" %s%s.log", $logarea,
$table);
           }
           else
           {
                #$control = sprintf (" %s%s_%d.ctl",
$ctlarea, $table, $idp);
                $log = sprintf (" %s%s_%d.log",
$logarea, $table, $idp);
           }
        }
        $cmd = sprintf ("%s control=%s%s%s
log=%s%s%s%s%s%s%s\n",

        $upwd,$control,$skip,$load,$log,$errors,$rows,

        $bsize,$silent,$dirbool,$parbool);
        &dump1("*load$node");
        &dump0("*wait") if ((!$parbool) && (!defined
$params{'load_no_waits'}));

# create the controlfiles (fixed-length fields or delimited
records)
        if ($params{'skip_mk_ldctlfs'} !~ /$table/)
        {
           &load_ctl_head;
           $infiles = 0;
           $badfiles = 0;
           $discardfiles = 0;
           $intopartitions = "BEGINNING"; # this will
collect the partition numbers that will be served.
           if ($dp < $npart) {
                if ($idp <= $remchild) {
                   $nchild = $numchild+1;
                }else{
                   $nchild = $numchild;
                }
           }else{
                if ($partcount <=
$sqlldr_per_partition_rest) {
                   $nchild = $infiles_per_sqlldr+1;
                }else{
                   $nchild = $infiles_per_sqlldr;
                }
           }
           for ($jpart=1; $jpart<=$nchild; $jpart++)
           {
                #print "ipart @ipart[$node]\n";
                $idbgpar = 0 if $idbgpar >= $dbgpar;
                #@ipart[$node] = int((($idp-
1)*$nchild+$jpart)/$dbgpar);
                #@ipart[$node] = @ipart[$node]+1 if
((($idp-1)*$nchild+$jpart)-@ipart[$node]*$dbgpar)>0);
                if ($dgp) # load in partition mode
                {
                   print @ipart."\n";
                   $pn=@ipart[$node];
                   $partitionname = $pname.$pn;
                   if (!($intopartitions =~
/$partitionname/)) {
                        if ($intopartitions =~
/BEGINNING/)
                        {
                           $intopartitions =
$partitionname;
                        }
                        else {
                           $intopartitions .=
",".$partitionname;}
                   }
#lvaz : 03/20/01
#need to extract the correct path for the flatfile
                @dplist_ff  = split(/,/,
$params{'tab_'.$table.'_flatfile_area'});
                $inff_path=$ff_path;
```

```
                if (defined
$params{'tab_'.$table.'_flatfile_area'})
                {
                  $off=($idbgpar);
                  for ($i_cnt=0; $i_cnt<=$off; $i_cnt++)
                  {
                    $ff_temp = shift @dplist_ff;
                  }
                  $inff_path =
$ff_temp.$params{'load_dbgen_'.$table.'_output_prefix'};
                }
                        @infile[$infiles++] = """ . $inff_path
. "_" . ++$idbgpar .
#lvaz : 03/20/01
                                "." . @ipart[$node] . """;
                        @badfile[$badfiles++] = """ .
$ff_path . "_" . $idbgpar .
                                "_" . @ipart[$node] . ".bad"";
                        @discardfile[$discardfiles++] = """ .
$ff_path . "_" . $idbgpar .
                                "_" . @ipart[$node] . ".dsc"";
                        #if ($remchild-- > 0)
                        #{
                        #    @infile[$infiles++] = """ .
$ff_path . "_" . ++$idbgpar .
                                #            "." . @ipart[$node] .
""";
                        #    @badfile[$badfiles++] = """ .
$ff_path . "_" . $idbgpar .
                        #        "_" . @ipart[$node] . ".bad"";
                        #    @discardfile[$discardfiles++] =
""" . $ff_path . "_" . $idbgpar .
                        #        "_" . @ipart[$node] . ".dsc"";
                        #}
                      }
                      elsif ($dbgpar == 1)
                      {
                        @infile[$infiles++] = """ . $ff_path .
".tbl"";
                        @badfile[$badfiles++] = """ .
$ff_path . ".bad"";
                        @discardfile[$discardfiles++] = """ .
$ff_path . ".dsc"";
                      }
                      else
                      {
                        @infile[$infiles++] = """ . $ff_path .
".tbl." . ++$idbgpar . """;
                        @badfile[$badfiles++] = """ .
$ff_path . "_" . $idbgpar . ".bad"";
                        @discardfile[$discardfiles++] = """ .
$ff_path . "_" . $idbgpar .
                                ".dsc"";
                        if ($remchild-- > 0)
                        {
                                @infile[$infiles++] = """ .
$ff_path . ".tbl." . ++$idbgpar . """;
                                @badfile[$badfiles++] = """ .
$ff_path . "_" . $idbgpar . ".bad"";
                                @discardfile[$discardfiles++]
= """ . $ff_path . "_" . $idbgpar .
                                        ".dsc"";
                        }
                      }
                      #print "vorher jpart $jpart ipart
@ipart[$node] iipart @iipart[$node] dbgpar $dbgpar\n";
                      if (@iipart[$node] >= $dbgpar){
                        @iipart[$node]=0;
                        @ipart[$node]=@ipart[$node]+1;
                      }
                      @iipart[$node]=@iipart[$node]+1;
                      #print "nacher jpart $jpart ipart
@ipart[$node] iipart @iipart[$node] dbgpar $dbgpar\n";
                }
                --$infiles;
                if ($up)
                {
                        print CTLFILE "INFILE ";
                        for ($fpart=0; $fpart<=$infiles;
$fpart++)
                        {
                          print CTLFILE @infile[$fpart] . ", "
if $fpart < $infiles;
                          print CTLFILE @infile[$fpart] . " " if
$fpart == $infiles;
                        }
                        print CTLFILE "BADFILE " .
@badfile[0] . " ";
                        print CTLFILE "DISCARDFILE " .
@discardfile[0] . "\n";
                }
                else
                {
                        for ($fpart=0; $fpart<=$infiles;
$fpart++)
                        {
                          print CTLFILE "INFILE " .
@infile[$fpart] . " ";
                          print CTLFILE "BADFILE " .
@badfile[$fpart] . " ";
                          print CTLFILE "DISCARDFILE " .
@discardfile[$fpart] . "\n";
                        }
                }
                print CTLFILE
"$params{'load_insert_type'}\n";
                print CTLFILE "into table $table\n";
                if ($dgp)
                {
                        #print CTLFILE "partition
($pname@ipart[$node])\n" if
!($params{'tab_'.$table.'_parttype'} =~ /hash/);
                        print CTLFILE "partition
($intopartitions)\n" if !($params{'tab_'.$table.'_parttype'}
=~ /hash/);
```

```
            }
            &load_ctl_tail;
        }
        $partcount = $partcount + 1;
    }
}
    &dump0("*wait") if !defined $params{'load_no_waits'}
&& $dgp;
    &time0("End of load for Partition: @ipart[$node] for
Table: $table") if ($dgp);

    &dump0("*wait") if !defined
$params{'load_no_waits'};
    &time0("End $table load");
}



sub dapop_frank ## 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/) ? ($ud = 1) : ($ud =
0);
    $params{'load_tables'} = $params{'tab_tables'} if
!defined $params{'load_tables'};
    @load_tablelist = split(/,/, $params{'load_tables'});
    foreach $table (@load_tablelist)
    {
        &time0("Begin $table load");

      $curts = $params{'tab_'.$table.'_ts'};
      $dp = $params{'tab_'.$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

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

      $params{'tab_'.$table.'_load_ctlf'} = $table.'.ctl' if
!defined $params{'tab_'.$table.'_load_ctlf'};
      @ctlf = split(/,/, $params{'tab_'.$table.'_load_ctlf'});
      $err = "ctl" 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;
      @datf = split(/,/,
$params{'tab_'.$table.'_load_datf'});
      $err = "dat" if ((@datf > 1) && (@datf != $dp));
      $dafexist = 1 if @datf > 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'} = "";
                $numts =
$params{'tab_'.$table.'_#part'};
                $numfil =
$params{$params{'tab_'.$table.'_ts'}.'_#files'};

                for ($j = 1; $j <= $numts; $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 <= $numts; $j++)
                           {
                              $tspnam =
sprintf("%s%d",$tsnam,$j);
                                ((($i == $numfil) && ($j ==
$numts)) ?
                                   ($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'})
                                {
                           $nextfile =
sprintf("\\\\\\\\\.\\\\%s%d_%d%s",$params{'tab_'.$table.'_p
art_lfn'},$j,$i,$addcomma);
                                }
                                    else
                                    {
                                 $nextfile =
sprintf("\\\\\\\\\.\\\\%s_%d%s",$tspnam,$i,$addcomma);
                           }
                                }
                               $params{'tab_'.$table.'_load_filf'}
.= $nextfile;
                        }
                   }
                }
                else
                {
                  $params{'tab_'.$table.'_load_filf'} =
$params{$curts.'_datafiles'};
                }
             }
             elsif ($params{'tab_'.$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
                           {
# Temporay workaround of 804 and SQLLDR bugs
#                             $nextfile =
sprintf("%s%s%s",@tsarea[$i],$dfile[$i],$addcomma);
                             $nextfile =
sprintf("\\\\\\\\\.\\\\%s%s",$dfile[$i],$addcomma);

                           }
                           $params{'tab_'.$table.'_load_filf'} .=
$nextfile;
                        }
                }

## 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;
      }
      $usels = 1 if (((($dp > 1) && (@datf == 1) &&
$params{'tab_'.$table.'_load_datf'} !~ /\#/));

# 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 (@datf > 1)
         {
            @daf[$i] = @datf[$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
                  # sqlldr.

                  if (($params{'tab_'.$table.'_#part'} > 0)
&&
                     ($dp >
$params{'tab_'.$table.'_#part'})) {
                     if ($dp %
$params{'tab_'.$table.'_#part'} == 0) {
                        if (@datf[0] =~ /\#.*\#/) {
                           # set number

$snum=int($i/$params{'tab_'.$table.'_#part'}) + 1;
                           # partiton number

$pnum=$i%$params{'tab_'.$table.'_#part'} + 1;
                           @daf[$i] = @datf[0];
                           @daf[$i] =~ s/\#/$snum/;
                           @daf[$i] =~ s/\#/$pnum/;
                        } else {
                           @daf[$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 $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/\#/$iplus/g;
        }
        if (@filf > 1)
        {
           @ff[$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_'.$table.'_ts'}.'_#files'};
#MP             print "$table \n";
#MP             print
"$params{$params{'tab_'.$table.'_ts'}.'_#files'}\n";
                $filnum = $i % $numfil + 1;
                @ff[$i] =~ s/\#/$filnum/g;
           } else {
                @ff[$i] =~ s/\#/$iplus/g;
           }
        }
     }

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

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

        open (CTLFILEH, ">$controlh");
        open (CTLFILET, ">$controlt");
        print CTLFILEH "---\n";
        if (!($pre72) && defined
$params{'tab_'.$table.'_loadextent'})
        {
                print CTLFILEH "options\n";
          print CTLFILEH "(\n";
            print CTLFILEH "storage = (initial
$params{'tab_'.$table.'_loadextent'} next
$params{'tab_'.$table.'_loadextent'})\n";
            print CTLFILEH ")\n";
          }

#       print CTLFILE "unrecoverable\n" if
($params{'load_unrecoverable'} =~ /[tT][rR][uU][eE]/);
          print CTLFILEH "unrecoverable\n" if
($params{'load_unrecoverable'} =~ /[tT][rR][uU][eE]/);

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



          print CTLFILET "-- This is where PARTITION
is specified.\n";
          print CTLFILET
"$params{'load_insert_type'}\n";
          print CTLFILET "fields terminated by
$params{'load_field_terminator'}\n" if ($ud == 1);
          print CTLFILET "(\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);
             $ctlline = sprintf ("  %-20s %s
%s$addcomma", $col, $pos,
$params{'tab_'.$table.'_'.$col.'_loadcolx'});
             print CTLFILET "$ctlline\n";
          }
          print CTLFILET ")\n";
          close (CTLFILEH);
          close (CTLFILET);

# create the controlfiles (fixed-length fields or delimited
records)
     if ($params{'skip_mk_ldctlfs'} !~ /$table/)
     {
             #print "control $control\n";
        open (CTLFILE, ">$control");
        print CTLFILE "---\n";
        print CTLFILE "---  $table.ctl for delimited
records\n" if ($ud == 1);
        print CTLFILE "---  $table.ctl for fixed-length
fields\n" if ($ud == 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
($params{'load_unrecoverable'} =~ /[tT][rR][uU][eE]/);
        print CTLFILE "load\n";
#         print CTLFILE "$params{'load_insert_type'}\n";
        print CTLFILE "into table $table\n";

        print CTLFILE "$params{'load_insert_type'}\n";
        print CTLFILE "fields terminated by
$params{'load_field_terminator'}\n" if ($ud == 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 ($ud == 0);
            $ctlline = sprintf ("  %-20s %s %s$addcomma",
$col, $pos, $params{'tab_'.$table.'_'.$col.'_loadcolx'});
            print CTLFILE "$ctlline\n";
        }
        print CTLFILE ")\n";
        close (CTLFILE);
    }

# 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;
        }

## NT Port (use dbgen to create flat files)
##
##         # if using pipes, create the pipes
##
##         if($up)
##         {
##            for ($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'});
##                    $nperset = int($dbgpar / $nset);
##            }
##
##            for ($i=0;$i<$dbgpar;$i++) {
##                    $cmd = $params{'load_flatfile_area'} .
"\n";
##                    $cmd2 = 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 =~ /\#/) {
##                              $snum = int($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 backgroud from here
##                    $cmd2 = " -f " . $cmd2;
##                    $cmd2 .= " & ";
##
##                    $cmd2 .= "\n";
##                    &dump2("*dbgen");
##            }
##         }
##
##         &dump0("*wait");
## NT End

        for ($i=0; $i < $dp; $i++)
        {
           &advmulti();
                $control = "";
                $log = "";
                $bad = "";
                $data="";
                $discard = "";
           $control = sprintf (" control=%s%s",
$params{'load_controlfile_area'}, @cf[$i]);
           $control = sprintf (" control=/host%s%s",
$params{'load_controlfile_area'}, @cf[$i]) if
($params{'special_machine'} eq 'ncube');
           $log = sprintf (" log=%s%s",
$params{'load_otherfile_area'}, @lf[$i]) if ($lfexist ==
1);
           $bad = sprintf (" bad=%s%s",
$params{'load_otherfile_area'}, @bf[$i]) if ($bfexist ==
1);
           $data = sprintf (" data=%s%s",
$params{'load_flatfile_area'}, @daf[$i]) if ($dafexist==
1);
           $discard = sprintf (" discard=%s%s",
$params{'load_otherfile_area'}, @dif[$i]) if ($difexist==
1);
                if ($ffexist ==1)
                {
              if ($curts == "")
                   {
                        $file = sprintf (" file=%s", @ff[$i]);
                   }
                   else
                   {
                $file = sprintf (" file=%s%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 = $split;
          }
##             if ($up)
##             {
##                     $load = " load=99999999";
##             }

          $cmd = sprintf
("%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s%s\n",$up
wd,$control,$log,$bad,$data,$discard,$dismax,$skip,$loa
d,$errors,$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");
##--       }
       &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");
   foreach $index (@indexlist)
   {
          if (defined $params{'ind_'.$index.'_table'})
          {
             $ob_type = 'table';
             $ob_stype = 'tab';
          }
          else
          {
             $ob_type = 'view';
             $ob_stype = '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'} =~
/[tT][rR][uU][eE]/)
          {
             $cmd .= "unique ";
          } elsif ($params{'ind_'.$index.'_bitmap'} =~
/[tT][rR][uU][eE]/) {
             $cmd .= "bitmap ";
          }
          $cmd .= "index $index\n";
       if (defined $params{'ind_'.$index.'_'.$ob_type})
       {
          $cmd .= "on $params{'ind_'.$index.'_'.$ob_type}
($params{'ind_'.$index.'_'.$ob_stype.'cols'})\n";
             $indtab =
$params{'ind_'.$index.'_'.$ob_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_'.$index.'_unrecoverable'} =~
/[tT][rR][uU][eE]/);
          } elsif ($params{'compatible'} =~ /8\./) {
             $cmd .= "nologging\n" if
($params{'ind_'.$index.'_nolg'} =~ /[tT][rR][uU][eE]/);
          }
          $cmd .= "compute statistics\n" if
($params{'ind_'.$index.'_compstats'} =~
/[tT][rR][uU][eE]/);
       $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'} =~ /[tT][rR][uU][eE]/);
      $cmd .= "nosort\n" if
($params{'ind_'.$index.'_nosort'} =~ /[tT][rR][uU][eE]/);

        if ((defined $params{'ind_'.$index.'_pardeg'}) ||
(defined $params{'ind-'.$index.'_parinst'}))
      {
         $cmd .= "parallel";
            if ($params{'ind_'.$index.'_pardeg'}
=~/[dD][eE][fF][aA][uU][lL]/)
            {
                  $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'} =~
/[lL][oO][cC][aA][lL]/)
         {
            $numpart =
$params{$ob_stype.'_'.$indtab.'_#part'};
            $cmd .= "local";
            if (defined
$params{'ind_'.$indtab.'_partnames'})
            {
                  &exp_ind_part_l;
            }
            else
            {
                  $cmd .= "\n";
            }
         } elsif ($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 .= ";\n";
      &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.'_table'} ";
         $cmd .= "modify (";
         while ($col = shift(@collist))
         {
            (@collist == 0) ? ($addcomma = "") :
($addcomma = ",");
            $cmd .= "$col
$params{'con_'.$const.'_constraint'}$addcomma";
         }
         $cmd .= ");\n";
      }
      else
      {
         $cmd .= "alter table
$params{'con_'.$const.'_table'} drop constraint $const;\n";
         $cmd .= "alter table
$params{'con_'.$const.'_table'} ";
         $cmd .= "add constraint $const\n";
         $cmd .= "      $params{'con_'.$const.'_constraint'}
key ($params{'con_'.$const.'_columns'}) ";
            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 {
                  $cmd .= ";\n";
            }
            if ($params{'con_'.$const.'_constraint'} =~
/foreign/) {
                  $cmd .= "alter table
$params{'con_'.$const.'_table'} ";
                  $cmd .= "enable novalidate primary
key;\n";
         }
            if ($params{'con_'.$const.'_constraint'} =~
/foreign/) {
                  $cmd .= "alter table
$params{'con_'.$const.'_table'} ";
                  $cmd .= "enable primary key";
            }
            if ($params{'con_'.$const.'_has_index'} =~
/[Tt][Rr][Uu][Ee]/) {
                  $cmd .= "";
            } else {
```

```
                    $cmd .= "using index\n";
                    $cmd .= "pctfree
$params{'con_'.$const.'_%f'}\n" if defined
$params{'con_'.$const.'_%f'};
                    $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/);
                    $cmd .= ";\n";
            }
        }
        &dump1("*sql");
            &dump0("*wait") if (($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 ixcre


sub exp_ind_part_l
{
    @ind_partnames = ();
    $params{'ind_'.$index.'_#part'} =
            $params{$ob_stype.'_'.$params{'ind_'.$index.'_'.
$ob_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);
            $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_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_partna
mes[$i]);
           } elsif ((@ind_partnames) &&
                    defined
$params{'ind_'.$index.'_part_ts'})
           {
                 $cmd .= 'tablespace ' . $ind_part_ts[$i] .
"\n";

        &process_part_storage('ind',$index,$ind_partna
mes[$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_l


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 ($i==$params{'ind_'.$table.'_#part'}-1) {
        $cmd .= "(MAXVALUE)\n";
      }
      else {
        $cmd .= "(" . $ind_part_vals[$i] . ")\n";
      }

         &process_part_params('ind','%f','pctfree',$index,
$ind_partnames[$i]);
         &process_part_params('ind','%u','pctused',$index
,$ind_partnames[$i]);

&process_part_params('ind','it','initrans',$index,$ind_partn
ames[$i]);

&process_part_params('ind','mt','maxtrans',$index,$ind_p
artnames[$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]);
         } elsif (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]/);
         } 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";
   }
   $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 (($params{$ob_stype.'_'.$indtab.'_'.$col.'_type'}
=~ /[cC][hH][aA][rR]/) ||
            ($params{$ob_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 $params{$ob_stype.'_'.$table.'_#part'})
          {
            $numpart =
$params{$ob_stype.'_'.$table.'_#part'};
          }
          else
          {
            $numpart = $params{'ind_'.$index.'_#part'};
          }
          $incre = $high_l_orderkey / $numpart;
          for ($i=0; $i < $numpart-1; $i++) {
            $pval = $pval + $incre;
            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 $params{$ob_stype.'_'.$table.'_#part'})
            {
              $numpart =
$params{$ob_stype.'_'.$table.'_#part'};
            }
            else
            {
              $numpart = $params{'ind_'.$index.'_#part'};
            }
            $incre = $high_l_partkey / $numpart;
            for ($i=0; $i < $numpart-1; $i++) {
              $pval = $pval + $incre;
              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'})
        {
            printf "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{$ob_stype.'_'.$indtab.'_'.$col.'_type'} =~
/[dD][aA][tT][eE]/)
            {
                if ($i == $params{'ind_'.$index.'_#part'} - 1)
                {
                    $ind_part_vals[$i] .= "MAXVALUE";
                } else {
                    $nls_format = (defined
$params{$ob_stype.'_'.$indtab.'_'.$col.'_date_format'}) ?
$params{$ob_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;
                }
```

```
        }
         elsif
(!(($params{$ob_stype.'_'.$indtab.'_'.$col.'_type'} =~
/[iI][nN][tT][eE][gG][eE][rR]/) ||
($params{$ob_stype.'_'.$indtab.'_'.$col.'_type'} =~
/[nN][uU][mM][bB][eE][rR]/)))
            {
            if ($index=~/l_ored/) {
              print "Before:$ind_part_vals[$i]";
            }
            $ind_part_vals[$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 anlyz
{

&dump0("#####################################
########################################");
   &dump0("# Analyze Phase");
   &time0("Begin anlyze");

   if ($params{'analyze_type'} =~ /via package
dbms.stats/)
   {
         &time0("Begin via package dbms.stats
analyzing");
            foreach $object (@anlyzlist)
            {
               $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{$phead.'_anl_type'}_".$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/);
               $cmd .= ");\n";
               &dump1("*sql");
            &dump0("*wait") if ($object =~ /schema/);
            }
            &time0("End per schema analyzing");
   }
   else
   {
            foreach $object (@anlyzlist)
            {
            &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%d%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++)
                        {
                                $filenm =~ s/\#/$i/g;
                                ($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
                        {
                                $cmd .= "compute
statistics;\n";
                        }
                        &dump1("*sql");
                    }
                } else {
                        if (($params{'anl_'.$object.'_type'} =~
/index/) &&
                                (defined
$params{'ind_'.$object.'_partition'})) {
                                $ind_entry = 'ind_'.$object;
                                if
($params{'ind_'.$object.'_partition'} =~
/[lL][oO][cC][aA][lL]/) {
                                        $params{$ind_entry.'_#part'}
=

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

$params{$ind_entry.'_partnames'} =

                $params{$ind_entry.'_partnames'} . $nextfile;
                                        }
                                }

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

                                # if partnames is specified as
XXXX#, then expand
                                if ($ind_partnames[0] =~ /\#/)
                                {
                                        $filenm =
shift(@ind_partnames);
                                        $savename = $filenm;

                $params{$ind_entry.'_partnames'} = "";

                                        for ($i = 1; $i <=
$params{$ind_entry.'_#part'}; $i++)
                                        {
                                           $filenm =~ s/\#/$i/g;

($i==$params{$ind_entry.'_#part'}) ? ($addcomma = "") :
                                                ($addcomma = ",");

$params{$ind_entry.'_partnames'} =

                $params{$ind_entry.'_partnames'} . $filenm .
```

```
                                $addcomma;
                            $filenm = $savename;
                        }

        @tab_partnames=split(/,/,$params{$ind_entry.'_
partnames'});
                        printf("Expanded $savename
to:\n$params{$ind_entry.'_partnames'}\n\n") if $verbose;
                    } else {
                        if (@ind_partnames !=
$params{$ind_entry.'_#part'})
                        {
                            print "Number of partitions
$params{$ind_entry.'_#part'} for $table doesn't match\n
_partnames parameter for
$params{$ind_entry.'_partnames'}\n";
                            exit(-1);
                        }
                    }
                    @ind_partnames =
split(/,/,$params{'ind_'.$object.'_partnames'});
                    foreach $partname (@ind_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";
                        }
                        &dump1("*sql");
                    }
                } else {
                    $cmd = "connect
$params{'user'}/$params{'passwd'};\n";
#       $cmd .= "analyze $params{'anl_'.$object.'_type'}
$object ";
#       $cmd .= "delete statistics;\n";
                    $cmd .= "analyze
$params{'anl_'.$object.'_type'} $object ";
                    if (defined
$params{'anl_'.$object.'_estimate'})
                    {
                        $cmd .= "estimate statistics
$params{'anl_'.$object.'_estimate'};\n";
                    }
                    else
                    {
                        $cmd .= "compute
statistics;\n";
                    }
                    &dump1("*sql");
                }
            }
            &time0("Done analyzing $object");
        }
        &time0("End anlyze");
    }
}

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/) {
        $cmd .= "alter table $object storage (next
$chstorlist{$object} pctincrease 0);\n";
      }
      else {
        $cmd .= "alter index $object storage (next
$chstorlist{$object} pctincrease 0);\n";
      }
    }
    &dump1("*sql");
}

sub chob
{

&dump0("#################################b###
########################################");
    &dump0("# Make CHOB Phase");

    if (@choblist) {
        $cmd = "connect
$params{'user'}/$params{'passwd'};\n";
```

```
   }
 FLOOP:
   while (@choblist)
   {
          $object = shift(@choblist);
          if ((defined
$params{'chob_tab_'.$object.'_degpar'}) || (defined
$params{'chob_tab'.$object.'_storage'}))
          { $objecttype = 'table';  $objecttypes =
'tab';$objectn=$object;}
          elsif ((defined
$params{'chob_ind_'.$object.'_degpar'}) || (defined
$params{'chob_ind'.$object.'_storage'}))
          {$objecttype = 'index'; $objecttypes =
'ind';$objectn=$object;}
          elsif ((defined
$params{'chob_viewlog_'.$object.'_degpar'}) || (defined
$params{'chob_viewlog'.$object.'_storage'}))
          {$objecttype = 'table'; $objecttypes =
'viewlog';$objectn="MLOG\\\$_".$object}
          else
          {next FLOOP;}
          if (defined
$params{'chob_'.$objecttypes.'_'.$object.'_degpar'})
          {
             $cmd .= "alter $objecttype $objectn";
             if
($params{'chob_'.$objecttypes.'_'.$object.'_degpar'} =~
/[dD][eE][fF][aA][uU][lL][tT]/)
             {
                $cmd .= " parallel";
             }
             elsif
($params{'chob_'.$objecttypes.'_'.$object.'_degpar'} =~
/[nN][oO][pP][aA][rR][aA][lL][lL][eE][lL]/)
             {
                $cmd .= " noparallel";
             }
             else
             {
                $cmd .= " parallel
$params{'chob_'.$objecttypes.'_'.$object.'_degpar'}"
                   if (defined
$params{'chob_'.$objecttypes.'_'.$object.'_degpar'});
             }
             $cmd .= ";\n";
          }
          if (defined
$params{'chob_'.$objecttypes.'_'.$object.'_storage'})
          {
             $cmd .= "alter $objecttype $objectn";
             if (defined
$params{'chob_'.$objecttypes.'_'.$object.'_storage'})
             {
                $cmd .= " storage
($params{'chob_'.$objecttypes.'_'.$object.'_storage'});\n";
             }
          }
          if ($objecttypes eq 'tab')
          {
             delete $params{'chob_tab_'.$object.'_degpar'};
             delete $params{'chob_tab'.$object.'_storage'};
             push (@choblist,$object);
          }
   }
   &dump1("*sql");
}




sub expln
{

&dump0("#####################################
########################################");
   &dump0("# Explain Plan Phase");

   $params{'qry_explain_queries'} =
$params{'qry_all_queries'} if !defined
$params{'qry_explain_queries'};
   if ($params{'qry_explain_queries'})
   {
      $astr = "2_1";
      $astr = "2_".$params{'db_maxinstances'} if defined
$params{'db_maxinstances'};
      &alttabs($astr);
      @qrynames = split(/,/,
$params{'qry_explain_queries'});
      foreach $qname (@qrynames)
      {
         $cmd = "";
         $cmd .= "$params{'qry_area'}expl." . $qname .
"\n";
         $cmd2 = "";
         $cmd2 = $cmd2 . "connect
$params{'user'}/$params{'passwd'};\n";
         $cmd2 = $cmd2 . "set longwidth 360\n";
         if (defined $params{'qry_'.$qname.'_text'})
         {
            $cmd2 = $cmd2 . "drop table plan_table;\n";
            $cmd2 = $cmd2 . '@' .
"$params{'dd_sql_area'}" . "utlxplan.sql\n";
            $cmd2 = $cmd2 . "explain plan for\n";
            $cmd2 = $cmd2 .
"$params{'qry_'.$qname.'_text'}\n";
            $cmd2 = $cmd2 . "select level, rpad(lpad('
',2*(level-1)) || operation || ' ' || object_name || ' ' || options ||
' (' || object_node || ',' || cost || ',' || cardinality || ')',68)\n";
            $cmd2 = $cmd2 . "from plan_table\n";
            $cmd2 = $cmd2 . "connect by parent_id=prior
id start with id=0;\n";
            $cmd2 = $cmd2 . "select level, lpad(' ',2*(level-
1)) || operation || ' ' || object_name || ' ' || options || ' (' ||
object_node || ')',\n";
            $cmd2 = $cmd2 . "other_tag, other\n";
```

```
            $cmd2 = $cmd2 . "from plan_table\n";
            $cmd2 = $cmd2 . "connect by parent_id=prior
id start with id=0;\n";
          }
          else
          {
             if (defined $params{'qry_'.$qname.'_texte'})
             {
                @expln_texts = split(/,/,
$params{'qry_'.$qname.'_texte'});
                $text_index = 1;
                while ($expln_next = shift (@expln_texts))
                {
                 EXPLNLOOP:
                   while ($expln_next ne ('text'.$text_index))
                   {
                      if (!defined
$params{'qry_'.$qname.'_text'.$text_index})
                      {
                         print "\n** Missing text for $qname:
text$text_index\n";
                         last EXPLNLOOP;
                      }
                      $cmd2 = $cmd2 .
"$params{'qry_'.$qname.'_text'.$text_index}\n";
                      $text_index++;
                   }
                   $cmd2 = $cmd2 . "drop table
plan_table;\n";
                   $cmd2 = $cmd2 . '@' .
"$params{'dd_sql_area'}" . "utlxplan.sql\n";
                   $cmd2 = $cmd2 . "explain plan for\n";
                   $sqltext = 'qry_'.$qname.'_'.$expln_next;
                   $cmd2 = $cmd2 . "$params{$sqltext}\n";
                   $cmd2 = $cmd2 . "select level, rpad(lpad('
',2*(level-1)) || operation || ' ' || object_name || ' ' || options ||
' (' || object_node || ',' || cost || ',' || cardinality || ')',68)\n";
                   $cmd2 = $cmd2 . "from plan_table\n";
                   $cmd2 = $cmd2 . "connect by
parent_id=prior id start with id=0;\n";
                   $cmd2 = $cmd2 . "select level, lpad('
',2*(level-1)) || operation || ' ' || object_name || ' ' || options ||
' (' || object_node || ')',\n";
                   $cmd2 = $cmd2 . "other_tag, other\n";
                   $cmd2 = $cmd2 . "from plan_table\n";
                   $cmd2 = $cmd2 . "connect by
parent_id=prior id start with id=0;\n";
                   $text_index++;
                }
                while (defined
$params{'qry_'.$qname.'_text'.$text_index})
                {
                   $cmd2 = $cmd2 .
"$params{'qry_'.$qname.'_text'.$text_index}\n";
                   $text_index++;
                }
             }
             else
             {
                print "\n** No sql text or texte corresponding
to $qname\n";
             }
          }
          &dump2("*sql2");
       }
    }
}

sub query
{

&dump0("#####################################
########################################");
   &dump0("# Query Phase");

   $params{'qry_run_queries'} =
$params{'qry_all_queries'} if !defined
$params{'qry_run_queries'};
   if ($params{'qry_run_queries'})
   {
      $params{'qry_config'} =  "1_1" if !defined
$params{'qry_config'};
      @qconfs = split(/,/, $params{'qry_config'});
      foreach $qconf (@qconfs)
      {
         &alttabs($qconf);
         if ($params{'qry_separate_queries'} !~ /true/)
         {
            $cmd = "";
            $cmd .= "$params{'qry_area'}qry.all." . $qconf
. "\n";
            $cmd2 = "";
            $cmd2 = $cmd2 . "set timing on\n";
            $cmd2 = $cmd2 . "connect
$params{'user'}/$params{'passwd'};\n";
         }
         @qrynames = split(/,/,
$params{'qry_run_queries'});
         foreach $qname (@qrynames)
         {
            if ($params{'qry_separate_queries'} =~ /true/)
            {
               $cmd = "";
               $cmd .= "$params{'qry_area'}qry." . $qname
. "." . $qconf . "\n";
               $cmd2 = "";
               $cmd2 = $cmd2 . "set timing on\n";
               $cmd2 = $cmd2 . "connect
$params{'user'}/$params{'passwd'};\n";
            }
            for ($i = 0; $i < $params{'qry_number_trials'};
$i++)
            {
               if (defined $params{'qry_'.$qname.'_text'})
               {
```

```
                $cmd2 = $cmd2 .
"$params{'qry_'.$qname.'_text'}\n";
              }
              else
              {
                $text_index = 1;
               WTLOOP:
                while(1)
                {
                  if (defined
$params{'qry_'.$qname.'_text'.$text_index})
                  {
                    $cmd2 = $cmd2 .
"$params{'qry_'.$qname.'_text'.$text_index}\n";
                    $text_index++;
                  }
                  else
                  {
                    print "No sql text corresponding to
$qname\n" if ($text_index == 1);
                    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 #does not work for OPS systems!!
          #if you have an OPS system use startdb_old
{
    $initora=$params{'startupfile_'.$_[0]};
    $mounttype=$_[1];
    $cmd = "startup pfile=$initora $mounttype\n";
    &dump1("*sql");
}
sub startdb_old
{
# 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][sS][eE][rR]/)
   {                                    # user managed
temporary tablespace
         if ($params{$tsname.'_temporary'} =~
/[tT][rR][uU][eE]/)
            {                           # temporary
tablespace
               $cmd .= "drop tablespace $tsname
including contents;\n";
               $cmd .= "create tablespace $tsname
temporary\n";
            }
            else                # user managed permanent
tablespace
            {
               $cmd .= "drop tablespace $tsname
including contents;\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][rR][uU][eE]/)
            {                           # temporary
tablespace
               $cmd .= "drop tablespace $tsname
including contents;\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
               {
                  $cmd .= "uniform";
               }
            }
         else                # system managed permanent
tablespace
         {
            $cmd .= "drop tablespace $tsname including
contents;\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][rR][uU][eE]/);
    $cmd .= ";\n";
    &dump1("*sql");
}

sub add_ts_rollb
{
    $ts_undo_l = $_[0];
    @ts_datafiles = split(/,/,
$params{'ts_'.$ts_undo_l.'_datafiles'});
    $ts_datafile = shift (@ts_datafiles);
    $cmd .= "create tablespace ts_".$ts_undo_l."\n";
    $cmd .= "  datafile
'$params{'ts_undo_area'}$ts_datafile' size
$params{'ts_'.$ts_undo_l.'_first_size'}
$params{'ts_'.$ts_undo_l.'_options'};\n";
    &dump1("*sql");
    &dump0("*wait");
    &add_dfs("ts_$ts_undo_l");

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

}

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][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=35000000");
    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 ("%-25s = %s\n", $1, $2);
         print IOFILE $entry;
      }
      if (defined $params{'io_ifile'})
      {
         $entry = sprintf ("%-25s = %s\n", "ifile",
$params{'io_ifile'});
         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_objectlist = @_;
   $cmd .= "connect
$params{'user'}/$params{'passwd'};\n";
   foreach $object (@ob_objectlist)
   {
         &create_object;
   }
   &dump1("*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]\s[tT][aA][bB][lL][eE]/)))
   {
            $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.'_%t'}\n" if defined
$params{$ob_entry.'_%t'};
            $cmd .= "pctfree $params{$ob_entry.'_%f'}\n" if
defined $params{$ob_entry.'_%f'};
            $cmd .= "pctused $params{$ob_entry.'_%u'}\n"
if defined $params{$ob_entry.'_%u'};
            $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_%f'} " if defined
$params{$ob_entry.'_of_%f'};
               $cmd .= "pctused
$params{$ob_entry.'_of_%u'} " if defined
$params{$ob_entry.'_of_%u'};
               $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)
   {
            $cmd .= "enable row movement\n" if
defined($params{$ob_entry.'_parttype'});
            if ($params{$ob_entry.'_creation_type'} =~
/[dD][iI][rR][eE][cC][tT]/)
            {
               $cmd .= "BUILD
".$params{$ob_entry.'_build_when'}."\n" if (defined
$params{$ob_entry.'_build_when'});
               &add_refresh_clause($ob_entry);
            }
```

```
        if ($params{$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_s
elect'});
        $cmd .= "\n";
   }

   if (($ob_type cmp 'viewlog') == 0)
   {
        $cmd .= "with rowid\n";
        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";
        }
        $cmd .= "including new values\n";
   }

   if ($params{$ob_entry.'_creation_type'} =~
/[wW][iI][tT][hH]\s[tT][aA][bB][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]/))
        {
           $cmd .= "enable query rewrite\n";
        }
        elsif ($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_s
elect'});
   }
   $cmd .= ";\n";
}

sub add_columns
{
   while ($col = shift(@ob_collist))
   {
        (@ob_collist == 0) ? ($addcomma = "") :
($addcomma = ",");
        if (($ob_type cmp 'tab') == 0 )
        {
           $columntype =
$params{$ob_entry.'_'.$col.'_type'};
        }
        else
        {
           $columntype = "";
        }
        $nextline = sprintf ("    %-20s %s
%s$addcomma\n", $col, $columntype,
$params{$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/(, )|(,\t)/,\n /g;
   $as_select =~ s/from|FROM/\nFROM\n/g;
   $as_select =~ s/select|SELECT/SELECT\n/g;
   $as_select =~ s/where|WHERE/\nWHERE\n/g;
   $as_select =~ s/group by|GROUP BY/\nGROUP
BY\n/g;
   $as_select =~ s/ and | AND /\n AND \n/g;
   $as_select =~ s/ or | OR /\n OR \n/g;
   $cmd .= "as select\n";
   $cmd .= "/*".$hint."*/"."\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'};
      $cmd .= "pctused $params{'clu_'.$cluster.'_%u'}\n"
if defined $params{'clu_'.$cluster.'_%u'};
      $cmd .= "initrans $params{'clu_'.$cluster.'_it'}\n" if
defined $params{'clu_'.$cluster.'_it'};
      $cmd .= "maxtrans $params{'clu_'.$cluster.'_mt'}\n"
if defined $params{'clu_'.$cluster.'_mt'};
      $cmd .= "size $params{'clu_'.$cluster.'_size'}\n" if
defined $params{'clu_'.$cluster.'_size'};
      $cmd .= "tablespace $params{'clu_'.$cluster.'_ts'}\n"
if defined $params{'clu_'.$cluster.'_ts'};
      $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.'_pardeg'}) ||
(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)
      {
         $cluindex = "ind_".$cluster;
         $cmd .= "drop index $cluindex;\n";
         $cmd .= "create index $cluindex on cluster
$cluster;\n";
      }
   }
   &dump1("*sql");
}


sub expand_partitions

# *MP* parameterize procedure: changed $tab_entry into
$ob_entry 1. parameter $_[0]
#                              $table into $ob          2.
parameter $_[1]
#                        introduced $ob_type             3.
parameter $_[2]

{
   $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(\n";
          &expand_partitions_doit($ob_entry,$ob,$ob_typ
e);
   }
   elsif ($params{$ob_entry.'_parttype'} =~
/[hH][aA][sS][hH]/)
   {
          $cmd .= "partition by hash (" .
$params{$ob_entry.'_partcol'} . ")\n";
          $cmd .= "partitions ".
$params{$ob_entry.'_#part'} . "\n";
   }
   elsif ($params{$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(\n";
          &expand_partitions_doit($ob_entry,$ob,$ob_typ
e);
   }
}                                  # no partitioning


sub expand_partitions_doit

# *MP* parameterize procedure: changed $tab_entry into
$ob_entry 1. parameter $_[0]
#                                  $table into $ob          2.
parameter $_[1]
#                           introduced $ob_type             3.
parameter $_[2]

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

   &process_part_brys ($ob_entry,$ob);
   &process_part_ts ($ob_entry,$ob);

   # complete the partition statement

   for ($i=0; $i < $params{$ob_entry.'_#part'}; $i++)
   {
          $cmd .= "partition " . $ob_partnames[$i] . "
values less than ";
       if ($i==$params{$ob_entry.'_#part'}-1) {
         $cmd .= "(MAXVALUE)\n";
       }
       else {
             $cmd .= "(" . $ob_part_vals[$i] . ")\n";
       }
```

```
        &process_part_params($ob_type,'%f','pctfree',$o
b,$ob_partnames[$i]);
        &process_part_params($ob_type,'it','initrans',$ob
,$ob_partnames[$i]);
        &process_part_params($ob_type,'mt','maxtrans',
$ob,$ob_partnames[$i]);

        if (defined
$params{$ob_entry.'_'.$ob_partnames[$i].'_ts'})
        {
           $cmd .= 'tablespace ' .
$params{$ob_entry.'_'.$ob_partnames[$i].'_ts'} . "\n";

&process_part_storage($ob_type,$ob,$ob_partnames[$i])
;
        } elsif (defined $params{$ob_entry.'_part_ts'}) {
           $cmd .= 'tablespace ' . $ob_part_ts[$i] . "\n";

&process_part_storage($ob_type,$ob,$ob_partnames[$i])
;
        }
        if (defined
$params{$ob_entry.'_'.$ob_partnames[$i].'_nolg'})
        {
           $cmd .= "nologging\n" if
($params{$ob_entry.'_'.$ob_partnames[$i].'_nolg'} =~
/[tT][rR][uU][eE]/);
        } elsif (defined
$params{$ob_entry.'_part_def_nolg'}) {
           $cmd .= "nologging\n" if
($params{$ob_entry.'_part_def_nolg'} =~
/[tT][rR][uU][eE]/);
        }
        $cmd .= (($i+1) ==
$params{$ob_entry.'_#part'}) ? "" : ",\n";
   }
   $cmd .= ")\n";
} #end expand_partitions_doit


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


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


sub process_part_ts

# *MP* parameterize procedure: changed 'tab'.$table into
$ob__entry 1. parameter $_[0] (same as $_entry in
expand_partition)
#                          $table into $ob_          2.
parameter $_[1] (same as $ob in expand_partition)

{
   $ob__entry = $_[0];
   $ob_ = $_[1];
   # is objecttype_<name>_part_ts is in the form of
XXXX#, expand it
   # else treat it as a comma separated list of ts names
   if (defined $params{$ob__entry.'_part_ts'})
   {
        $nts=$params{$ob__entry.'_part_ts'};
        $nts=~s/#//;
        $nts=$params{$nts."_instances"};
        @ob_part_ts =
split(/,/,$params{$ob__entry.'_part_ts'});
        if ($ob_part_ts[0] =~ /\#/)
        {
           $filenm = shift(@ob_part_ts);
           $savename = $filenm;
           $params{$ob__entry.'_part_ts'} = "";
           $its = 0;
           for ($i = 1; $i <=
$params{$ob__entry.'_#part'}; $i++)
           {
                $its = $its + 1;
                if ($its > $nts) {
                   $its=1;
                }
                $filenm =~ s/\#/$its/g;
                ($i == $params{$ob__entry.'_#part'}) ?
($addcomma = "") :
                   ($addcomma = ",");
                $params{$ob__entry.'_part_ts'} =
                   $params{$ob__entry.'_part_ts'} .
$filenm . $addcomma;
                $filenm = $savename;
           }
           @ob_part_ts =
split(/,/,$params{$ob__entry.'_part_ts'});
```

```
        } else {
            if (@ob_part_ts !=
$params{$ob__entry.'_#part'})
            {
                if (($params{$ob__entry.'_#part'} %
@ob_part_ts) == 0) {
                    $numfil = @ob_part_ts;
                    $p = $params{$ob__entry.'_#part'} /
$numfil;
                    $fil = "";
                    for ($i = 0; $i < $p; $i++)
                    {
                        if ($i == 0) {
                            $fil =
$params{$ob__entry.'_part_ts'};
                        } else {
                            $fil =
join(',',$fil,$params{$ob__entry.'_part_ts'});
                        }
                    }
                    @ob_part_ts = split(/,/,$fil);
                } else {
                    print "Number of partitions
$params{$ob__entry.'_#part'} for object $ob_\n doesn't
match $table_part_ts parameter.\n";
                    exit (-1);
                }
            }
        }
    }
} # end of process_part_ts


sub process_part_brys


# *MP* parameterize procedure: changed 'tab'.$table into
$ob__entry 1. parameter $_[0] (same as $ob_entry in
expand_partition)
#                               $table into $ob_          2.
parameter $_[1] (same as $ob in expand_partition)

{
    $ob__entry = $_[0];
    $ob_ = $_[1];

    $cnt = 0;
    @ob_part_vals = ();

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


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

        @ob_part_col =
split(/,/,$params{$ob__entry.'_'.$col.'_partvals'});
        if (@ob_part_col !=
$params{$ob__entry.'_#part'})
        {
            printf "Number of partition boundary values
%d for column $col in object $ob_ doesn't match the
number of partitions ($params{$ob__entry.'_#part'}) of
the object\n", ($#ob_part_col + 1);
            exit(-1);
        }
        for ($i=0; $i < $params{$ob__entry.'_#part'};
$i++)
        {
            ($cnt == $#pcollist) ? ($addcomma = "") :
($addcomma = ",");
            if ($ob_part_col[$i] =~
/[Mm][Aa][Xx][Vv][Aa][Ll][Uu][Ee]/) {
                $addquote = "";
            }
            if ($params{$ob__entry.'_'.$col.'_type'} =~
/[dD][aA][tT][eE]/)
            {
             if ($i == $params{$ob_entry.'_#part'} - 1)
             {
               $ob_part_vals[$i] .= "MAXVALUE";
             } else {
                    $nls_format = (defined
$params{$ob__entry.'_'.$col.'_date_format'}) ?
$params{$ob__entry.'_'.$col.'_date_format'} : "YYYY-
MM-DD";
                    $ob_part_vals[$i] .=
"to_date('".$ob_part_col[$i]."','";
                    $ob_part_vals[$i] .=
$nls_format."')".$addcomma;
             }
            } else {
                $ob_part_vals[$i] = $ob_part_vals[$i] .
$addquote.
                  $ob_part_col[$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";
   }
   &dump1("*sql");
   &dump0("*wait");
}

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

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
{
   print "control $control\n";
   open (CTLFILE, ">$control");
   print CTLFILE "---\n";
   print CTLFILE "---  $table.ctl for delimited records\n"
if ($ud == 1);
   print CTLFILE "---  $table.ctl for fixed-length fields\n"
if ($ud == 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
$params{'load_field_terminator'}\n" if ($ud == 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 ($ud == 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 $bmtype)
   {
      eval(&$bmtype);
      &assigndefs;
   }
   &auxdefaults;
   &assigndefs;


$params{'dd_sql_area'}="$ENV{'ORACLE_HOME'}/rdb
ms/admin/" if !(defined $params{'dd_sql_area'});

$params{'dd_sqlplus_area'}="$ENV{'ORACLE_HOME'
}/sqlplus/admin/" if !(defined
$params{'dd_sqlplus_area'});

   if (defined $params{'compatible'})
   {
      $pre72 = 1 if $params{'compatible'} =~ /7\.(0|1)/;
      $pre73 = 1 if $params{'compatible'} =~ /7\.(0|1|2)/;
   }
   $params{'dbs_area'} =~
s/\?/$ENV{'ORACLE_HOME'}/g;
   $params{'load_tables'} = $params{'tab_tables'} if
!defined $params{'load_tables'};
   foreach $phase (@phases)
   {
      $params{$phase.'_num_inst'} =
$params{'def_num_inst'} if !defined
$params{$phase.'_num_inst'};
      $params{$phase.'_max_bg'} = $params{'max_bg'} if
!defined $params{$phase.'_max_bg'};
         $params{'startupfile_'.$phase} =
$params{'io_ifile'} if
!defined($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;
   }
   $params{'ops_nodes'} = 'undo' if
!defined($params{'ops_nodes'});
   @ops_nodes = split(/,/,$params{'ops_nodes'}) if
defined $params{'ops_nodes'};
   $params{'db_maxinstances'} = @ops_nodes if !defined
$params{'db_maxinstances'};

   #----------------------- undo tablespace definition ---------
-------------------------
   $params{'ts_undo_#rs'} = $params{'def_num_inst'} if
!defined $params{'ts_undo_#rs'};
```

```
   $params{'ts_undo_area'} = $params{'dbs_area'} if
!defined $params{'ts_undo_area'};
   $params{'ts_undo_rs_prefix'} = 'r' if !defined
$params{'ts_undo_rs_prefix'};
   $save_datafiles = $params{'ts_undo_datafiles'};

&expand_filename('ts_undo_datafiles',$params{'ts_undo_
#files'});

   if (@ops_nodes > 1)
   {
          foreach $node (@ops_nodes)
          {
             $params{'ts_undo_'.$node.'_#files'} =
$params{'ts_undo_#files'}
             if
!defined($params{'ts_undo_'.$node.'_#files'});

             $df= $save_datafiles.'_'.$node;

             $params{'ts_undo_'.$node.'_datafiles'} = $df
                   if
!defined($params{'ts_undo_'.$node.'_datafiles'});

&expand_filename('ts_undo_'.$node.'_datafiles',$params{'
ts_undo_'.$node.'_#files'});

             $params{'ts_undo_'.$node.'_first_size'} =
$params{'ts_undo_first_size'}
             if
!defined($params{'ts_undo_'.$node.'_first_size'});

             $params{'ts_undo_'.$node.'_#rs'} =
$params{'ts_undo_#rs'}
             if !defined($params{'ts_undo_'.$node.'_#rs'});

             $params{'ts_undo_'.$node.'_rs_storage'} =
$params{'ts_undo_rs_storage'}
             if
!defined($params{'ts_undo_'.$node.'_rs_storage'});

             $pr = $params{'ts_undo_rs_prefix'}.$node;
             $params{'ts_undo_'.$node.'_rs_prefix'} = $pr
                   if
!defined($params{'ts_undo_'.$node.'_rs_prefix'});
          }
   }

   $params{'ts_log_#files'} = $params{'ts_log_#threads'}
* $params{'ts_log_#files_pt'};
   $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'};
             }
          }
   }

   @ts_data = split(/,/, $params{'ts_data_names'});
   @ts_index = split(/,/, $params{'ts_index_names'});
   @ts_temp = split(/,/, $params{'ts_temp_names'});

# expand tablespaces
```

```
   @ts_data_new =();
   $ts_data_names="@@";
   foreach $ts_entry (@ts_data)
   {
           if (defined($params{$ts_entry.'_instances'})){
              for ($i = $params{$ts_entry.'_instances'}; $i >
0; $i --) {
                      $ts_new_entry = $ts_entry.$i;

           $params{$ts_new_entry.'_size'}=$params{$ts_e
ntry.'_size'};

           $params{$ts_new_entry.'_datafiles'}=$params{$
ts_entry.'_datafiles'};

           $params{$ts_new_entry.'_datafiles'}=~s/\#/$i/;

           $params{$ts_new_entry.'_#files'}=$params{$ts_
entry.'_#files'};

           $params{$ts_new_entry.'_storage'}=$params{$ts
_entry.'_storage'};
                      unshift (@ts_data_new, $ts_new_entry);

           $ts_data_names=$ts_data_names.",".$ts_new_en
try;
              }
           }
           else {
              unshift (@ts_data_new, $ts_entry);

$ts_data_names=$ts_data_names.",".$ts_entry;
           }
   }
   $ts_data_names=~s/@@,//;
   #print "ts_data_names: $ts_data_names\n";
   @ts_data = @ts_data_new;
   #print "ts_all before substitution @ts_all\n";
   #print "ts_all_new before substitution @ts_all_new\n";
   #while (@ts_data) {
#          $ts = shift (@ts_data);
           #unshift (@ts_data_new,$ts);
   #}
   #foreach $ts_entry (@ts_data)
   #{
#          print "ts_entry: $ts_entry\n";
#          print "size: $params{$ts_entry.'_size'}\n";
#          print "datafiles:
$params{$ts_entry.'_datafiles'}\n";
#          print "#files: $params{$ts_entry.'_#files'}\n";
#          print "storage: $params{$ts_entry.'_storage'}\n";
#   }

   #print "ts_all after substitution @ts_all\n";
   #print "$ts_list\n";
   #exit(0);

   $mostts =
"$params{'ts_temp_names'},$ts_data_names,$params{'ts_
index_names'}";
   $allts =
"ts_sys,ts_log,$params{'ts_undo'},$params{'ts_temp_nam
es'},$ts_data_names,$params{'ts_index_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'};
   }



   #rint "ts_all: @ts_all\n";
   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 $savename
to:\n$params{$ts_entry.'_datafiles'}\n\n") if $verbose;
      }
      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)
   {
        $params{'load_dbgen_'.$table.'_option_C'} =
$params{'load_dbgen_def_option_C'} if !defined
$params{'load_dbgen_'.$table.'_option_C'};

        $params{'tab_'.$table.'_ts'} = "ts_data" if
!defined $params{'tab_'.$table.'_ts'};
        if (defined($params{'load_deg_parallel'})) {
           $params{'tab_'.$table.'_load_degpar'} =
$params{'load_deg_parallel'} if !defined
$params{'tab_'.$table.'_load_degpar'};
        } else {
           $params{'tab_'.$table.'_load_degpar'} = 1 if
!defined $params{'tab_'.$table.'_load_degpar'};
        }

        #$params{'load_dbgen_'.$table.'_output_prefix'}
= "$table.tbl" if !defined
$params{'load_dbgen_'.$table.'_output_prefix'};

        $params{'tab_'.$table.'_load_datf'} = "$table.tbl"
if !defined $params{'tab_'.$table.'_load_datf'};

        $params{'tab_'.$table.'_load_badf'} =
"$table.badf" if !defined
$params{'tab_'.$table.'_load_badf'};

        $params{'tab_'.$table.'_load_ctlf'} = "$table.ctl"
if !defined $params{'tab_'.$table.'_load_ctlf'};

        $params{'tab_'.$table.'_load_logf'} =
"$table.log" if !defined
$params{'tab_'.$table.'_load_logf'};

        $params{'tab_'.$table.'_load_direct'} = "true" if
!defined $params{'tab_'.$table.'_load_direct'};

        $params{'tab_'.$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'};
        $params{$ob_entry.'_rewrite'} = "" if !defined
$params{$ob_entry.'_rewrite'};
   }

   @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'});
```

```
   @anlyzlist = split(/,/, $params{'anl_objects'});
   @anlyzlist = "schema" if (@anlyzlist == 0);
   foreach $object (@anlyzlist)
   {
          $params{'anl_'.$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;}
   $params{'plcre_recreate_extended_partitions'} =
"false" if !defined
($params{'plcre_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, 'dbs_area=/tmp');
   push (@defparams, 'dd_sql_area=?/rdbms/admin/');
   push (@defparams, 'max_bg=-1');
   push (@defparams, 'user=bmuser');
   push (@defparams, 'passwd=bmpasswd');
   push (@defparams, 'load_type=delim');
   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,query
rewrite,global query rewrite');
   push (@defparams, 'privileges=DBA');
   push (@defparams, 'qry_number_trials=1');
}

sub tpcd
{
   @defparams = ();
   push (@defparams, 'user=tpcd');
   push (@defparams, 'passwd=tpcd');
   push (@defparams,
'tab_tables=lineitem,orders,partsupp,parts,customer,suppli
er,nation,region');
   push (@defparams, 'view_views=mav_li,mv_locs');
   push (@defparams, 'viewlog_list=lineitem,orders');
   push (@defparams,
'mav_li_columns=count_p_group,l_shipdate,l_returnflag,l
_linestatus,sum_qty,sum_base_price ,sum_disc_price
,count_disc_price,sum_charge,
count_charge,count_qty,count_price,sum_disc,count_disc,
count_order');
   push (@defparams,
'tab_lineitem_columns=l_orderkey,l_partkey,l_suppkey,l_
linenumber,l_quantity,l_extendedprice,l_discount,l_tax,l_
returnflag,l_linestatus,l_shipdate,l_commitdate,l_receiptd
ate,l_shipinstruct,l_shipmode,l_comment');
   push (@defparams,
'tab_orders_columns=o_orderkey,o_custkey,o_orderstatus
,o_totalprice,o_orderdate,o_orderpriority,o_clerk,o_shipp
riority,o_comment');
```

```
    push (@defparams,
'tab_partsupp_columns=ps_partkey,ps_suppkey,ps_availqty,ps_supplycost,ps_comment');
    push (@defparams,
'tab_parts_columns=p_partkey,p_name,p_mfgr,p_brand,p_type,p_size,p_container,p_retailprice,p_comment');
    push (@defparams,
'tab_customer_columns=c_custkey,c_name,c_address,c_nationkey,c_phone,c_acctbal,c_mktsegment,c_comment');
    push (@defparams,
'tab_supplier_columns=s_suppkey,s_name,s_address,s_nationkey,s_phone,s_acctbal,s_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=char(25)');
    push (@defparams,
'tab_lineitem_l_shipmode_type=char(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=char(15)');
    push (@defparams,
'tab_orders_o_clerk_type=char(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=char(25)');
    push (@defparams,
'tab_parts_p_brand_type=char(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=char(10)');
    push (@defparams,
'tab_parts_p_retailprice_type=number');
    push (@defparams,
'tab_parts_p_comment_type=varchar(23)');
    push (@defparams, 'tab_parts_p_partkey_pos=1:12');
    push (@defparams, 'tab_parts_p_name_pos=13:67');
    push (@defparams, 'tab_parts_p_mfgr_pos=68:92');
    push (@defparams, 'tab_parts_p_brand_pos=93:102');
    push (@defparams, 'tab_parts_p_type_pos=103:127');
    push (@defparams, 'tab_parts_p_size_pos=128:139');
    push (@defparams,
'tab_parts_p_container_pos=140:149');
    push (@defparams,
'tab_parts_p_retailprice_pos=150:164');
    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=char(15)');
    push (@defparams,
'tab_customer_c_acctbal_type=number');
    push (@defparams,
'tab_customer_c_mktsegment_type=char(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=char(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=char(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=char(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=char(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_lineitem_output_prefix='.'lineitem');
    push (@defparams,
'load_dbgen_orders_output_prefix='.'orders');
    push (@defparams,
'load_dbgen_customer_output_prefix='.'customer');
    push (@defparams,
'load_dbgen_parts_output_prefix='.'parts');
    push (@defparams,
'load_dbgen_partsupp_output_prefix='.'partsupp');
    push (@defparams,
'load_dbgen_supplier_output_prefix='.'supplier');
    push (@defparams,
'load_dbgen_nation_output_prefix='.'nation');
    push (@defparams,
'load_dbgen_region_output_prefix='.'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,ten,twenty,
hundred,thousand,fivethous,tenthous,odd100,even100,stri
ngu1,stringu2,string4');
    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_odd100_type=number');
    push (@defparams, 'tab_wisc_even100_type=number');
    push (@defparams,
'tab_wisc_stringu1_type=varchar(52)');
    push (@defparams,
'tab_wisc_stringu2_type=varchar(52)');
    push (@defparams,
'tab_wisc_string4_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_odd100_loadcolx=integer external');
```

```
   push (@defparams,
'tab_wisc_even100_loadcolx=integer external');
   push (@defparams,
'tab_wisc_stringu1_loadcolx=char(52)');
   push (@defparams,
'tab_wisc_stringu2_loadcolx=char(52)');
   push (@defparams,
'tab_wisc_string4_loadcolx=char(52)');
}

sub expand_filename
{
   # 1. parameter: name of datafiles in associative array
(params)
   # 2. parameter: number of files the name should be
expanded to
   @tsdf = split(/,/, $params{$_[0]});
   $no_files = $_[1];
   $cntr = 1;
   if ($tsdf[0] =~ /\#/) # we want to replace all #'s
   {
          $filenm = shift(@tsdf);
          $savename = $filenm;
          $params{$_[0]} = "";
          for ($i = 0; $i < $no_files; $i++)
          {
             $filenm =~ s/\#/$cntr/g;
             $cntr++;
             (($i+1) == $no_files) ? ($addcomma = "") :
($addcomma = ",");
             $params{$_[0]} .= $filenm . $addcomma;
             $filenm = $savename;
          }
          printf ("Expanded $savename
to:\n$params{$_[1]}\n\n") if $verbose;
   }
   else
   {
          if (@tsdf != $no_files)
          {
             print "Number of files (@ts_datafiles) for
$_[1] doesn't match\n _datafile parameter $no_files}\n";
             exit(-1);
          }
   }
}

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 ("$pdfile");
   if (-e _)
   {
          open (PDFILE, "$pdfile");
          while (<PDFILE>)
          {
             ($key,$desc)=split (/::/,$_,2);
#            $key = shift(@l);
             $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
{
   $result = 1;
   return if ($runsilent == 1);
   @keys = keys %paramdesc;
   $p = $_[0]."!";
   while ($#keys >=0)
   {
          $key = pop(@keys);
          $key =$key."!";
          return if ($p =~ /$key/);
   }
   $result = -1;
}
```

Source Code

Bcd2.c

```
#ifdef RCSID
static char *RCSid =
  "$Header: bcd2.c 11-apr-2001.15:06:41 mpoess Exp $ ";
#endif /* RCSID */

/* Copyright (c) Oracle Corporation 1999. All Rights Reserved. */

/*

  NAME
    bcd2.c - <one-line expansion of the name>

  DESCRIPTION
    <short description of component this file declares/defines>

  PUBLIC FUNCTION(S)
    <list of external functions declared/defined - with one-line
descriptions>

  PRIVATE FUNCTION(S)
    <list of static functions defined in .c file - with one-line
descriptions>

  RETURNS
    <function return values, for .c file with single function>

  NOTES
    <other useful comments, qualifications, etc.>

  MODIFIED   (MM/DD/YY)
  mpoess     04/11/01 -
  mpoess     07/12/99 - exchange with new version July 8 1999
  mpoess     06/28/99 - creation
  mpoess     06/28/99 - Creation

*/


/* @(#)bcd2.c           2.1.8.1 */
/*
 * bcd.c: conversion routines for multi-byte arithmetic
 *
 * defined routines:
 * bin_bcd2(long binary, long *low_res, long *high_res)
 * bcd2_bin(long *dest, long bcd)
 * bcd2_add(long *bcd_low, long *bcd_high, long addend)
 * bcd2_sub(long *bcd_low, long *bcd_high, long subend)
 * bcd2_mul(long *bcd_low, long *bcd_high, long multiplier)
 * bcd2_div(long *bcd_low, long *bcd_high, long divisor)
 * long bcd2_mod(long *bcd_low, long *bcd_high, long modulo)
 * long bcd2_cmp(long *bcd_low, long *bcd_high, long compare)
 */
#include <stdio.h>
#include "bcd2.h"       /* for function prototypes */

#define DIGITS_PER_LONG 7
#define WORD_DIVISOR    10000000
#define GET_DIGIT(num, low, high) \
   ((num) >= DIGITS_PER_LONG)? \
      (high & (0xF << (4 * ((num) - DIGITS_PER_LONG)))) \
         >> (((num) - DIGITS_PER_LONG) * 4): \
      (low & (0xF << (4 * (num)))) >> ((num) * 4)
#define SET_DIGIT(value, num, low, high) \
   if ((num) >= DIGITS_PER_LONG) \
      { \
      *high &= \
         (0xFFFFFFF ^ (0xF << (4 * ((num) - DIGITS_PER_LONG)))); \
      *high |= (value << (4 * ((num) - DIGITS_PER_LONG))); \
      } \
   else \
      { \
      *low = (*low & (0xFFFFFFF ^ (0xF << (4 * (num))))); \
      *low |= (value << (4 * (num))); \
      }
int
bin_bcd2(long binary, long *low_res, long *high_res)
{
   char number[15],
        *current;
   int count;
   long *dest;

            *low_res = *high_res = 0;
   sprintf(number, "%014ld", binary);
   for (current = number, count=13; *current; current++, count--)
      {
      dest = (count < DIGITS_PER_LONG)?low_res:high_res;
      *dest = *dest << 4;
      *dest |= *current - '0';
      }
   return(0);
}

int
bcd2_bin(long *dest, long bcd)
{
   int count;
   long mask;

   count = DIGITS_PER_LONG - 1;
   mask = 0xF000000;
            *dest = 0;
   while (mask)
      {
      *dest *= 10;
      *dest += (bcd & mask) >> (4 * count);
      mask = mask >> 4;
      count -= 1;
      }
   return(0);
}

int
bcd2_add(long *bcd_low, long *bcd_high, long addend)
{
   long tmp_lo, tmp_hi, carry, res;
   int digit;

   bin_bcd2(addend, &tmp_lo, &tmp_hi);
   carry = 0;
   for (digit=0; digit < 14; digit++)
      {
      res = GET_DIGIT(digit, *bcd_low, *bcd_high);
      res += GET_DIGIT(digit, tmp_lo, tmp_hi);
      res += carry;
      carry = res / 10;
      res %= 10;
      SET_DIGIT(res, digit, bcd_low, bcd_high);
      }
   return(carry);
}

int
bcd2_sub(long *bcd_low, long *bcd_high, long subend)
{
   long tmp_lo, tmp_hi, carry, res;
   int digit;
```

```
    bin_bcd2(subend, &tmp_lo, &tmp_hi);
    carry = 0;
    for (digit=0; digit < 14; digit++)
        {
        res = GET_DIGIT(digit, *bcd_low, *bcd_high);
        res -= GET_DIGIT(digit, tmp_lo, tmp_hi);
        res -= carry;
        if (res < 0)
                                        {
                                        res += 10;
                                        carry = 1;
                                        }
        SET_DIGIT(res, digit, bcd_low, bcd_high);
        }
    return(carry);
}

int
bcd2_mul(long *bcd_low, long *bcd_high, long multiplier)
{
    long tmp_lo, tmp_hi, carry, m_lo, m_hi, m1, m2;
    int udigit, ldigit, res;

    tmp_lo = *bcd_low;
    tmp_hi = *bcd_high;
    bin_bcd2(multiplier, &m_lo, &m_hi);
    *bcd_low = 0;
    *bcd_high = 0;
    carry = 0;
    for (ldigit=0; ldigit < 14; ldigit++)
        {
        m1 = GET_DIGIT(ldigit, m_lo, m_hi);
        carry = 0;
        for (udigit=0; udigit < 14; udigit++)
            {
            m2 = GET_DIGIT(udigit, tmp_lo, tmp_hi);
            res = m1 * m2;
            res += carry;
            if (udigit + ldigit < 14)
                {
                carry = GET_DIGIT(udigit + ldigit, *bcd_low, *bcd_high);
                res += carry;
                }
            carry = res / 10;
            res %= 10;
            if (udigit + ldigit < 14)
                SET_DIGIT(res, udigit + ldigit, bcd_low, bcd_high);
            }
        }
    return(carry);
}

int
bcd2_div(long *bcd_low, long *bcd_high, long divisor)
{
    long tmp_lo, tmp_hi, carry, d1, res, digit;


    carry = 0;
    tmp_lo = *bcd_low;
    tmp_hi = *bcd_high;
    *bcd_low = *bcd_high = 0;
    for (digit=13; digit >= 0; digit--)
        {
        d1 = GET_DIGIT(digit, tmp_lo, tmp_hi);
        d1 += 10 * carry;
        res = d1 / divisor;
        carry = d1 % divisor;
        SET_DIGIT(res, digit, bcd_low, bcd_high);
        }
    return(carry);
}
```

```
long
bcd2_mod(long *bcd_low, long *bcd_high, long modulo)
{
        long tmp_low, tmp_high;

        tmp_low = *bcd_low;
        tmp_high = *bcd_high;
        while (tmp_high || tmp_low > modulo)
                bcd2_sub(&tmp_low, &tmp_high, modulo);
        return(tmp_low);
}

long
bcd2_cmp(long *low1, long *high1, long comp)
{
        long temp = 0;

    bcd2_bin(&temp, *high1);
        if (temp > 214)
                return(1);
    bcd2_bin(&temp, *low1);
        return(temp - comp);
}

#ifdef TEST_BCD
#include <values.h>

main()
{
long bin, low_bcd, high_bcd;
int i;

bin = MAXINT;
printf("%ld\n", bin);
bin_bcd2(bin, &low_bcd, &high_bcd);
printf("%ld  %ld\n", high_bcd, low_bcd);
bin = 0;
bcd2_bin(&bin, high_bcd);
bcd2_bin(&bin, low_bcd);
printf( "%ld\n", bin);
for (i=9; i >= 0; i--)
    printf("%dth digit in %d is %d\n",
        i, bin, GET_DIGIT(i, low_bcd, high_bcd));
bcd2_add(&low_bcd, &high_bcd, MAXINT);
bin = 0;
bcd2_bin(&bin, high_bcd);
high_bcd = bin;
bin = 0;
bcd2_bin(&bin, low_bcd);
low_bcd = bin;
printf( "%ld%07ld\n", high_bcd, low_bcd);
bin_bcd2(14, &low_bcd, &high_bcd);
bcd2_mul(&low_bcd, &high_bcd, 23L);
bin = 0;
bcd2_bin(&bin, high_bcd);
bcd2_bin(&bin, low_bcd);
printf( "%ld\n", bin);
bcd2_div(&low_bcd, &high_bcd, 10L);
bin = 0;
bcd2_bin(&bin, high_bcd);
bcd2_bin(&bin, low_bcd);
printf( "%ld\n", bin);
}
#endif /* TEST */
```

bcd2.h

```
/*
 * $Header: bcd2.h 11-apr-2001.15:06:13 mpoess Exp $
 */

/* Copyright (c) Oracle Corporation 1999. All Rights Reserved. */

/* NOTE:  See 'header_template.doc' in the 'doc' dve under the 'forms'
      directory for the header file template that includes instructions.
*/

/*
  NAME
    <<<LOWERCASE_FILENAME>>>.h - <one-line expansion of the
name>

  DESCRIPTION
    <short description of component this file declares/defines>

  RELATED DOCUMENTS

  INSPECTION STATUS
    Inspection date:
    Inspection status:
    Estimated increasing cost defects per page:
    Rule sets:

  ACCEPTANCE REVIEW STATUS
    Review date:
    Review status:
    Reviewers:

  PUBLIC FUNCTION(S)
    <list of external functions declared/defined - with one-line
descriptions>

  PRIVATE FUNCTION(S)
    <list of static functions defined in .c file - with one-line
descriptions>

  EXAMPLES

  NOTES
    <other useful comments, qualifications, etc.>

  MODIFIED   (MM/DD/YY)
  mpoess      04/11/01 -
  mpoess      07/12/99 - exchange with new version July 8 1999
  mpoess      06/28/99 - creation
  mpoess      06/28/99 - Creation

*/
/*
 *  Sccsid:     @(#)bcd2.h            2.1.8.1
 */
int bin_bcd2(long binary, long *low_res, long *high_res);
int bcd2_bin(long *dest, long bcd);
int bcd2_add(long *bcd_low, long *bcd_high, long addend);
int bcd2_sub(long *bcd_low, long *bcd_high, long subend);
int bcd2_mul(long *bcd_low, long *bcd_high, long multiplier);
int bcd2_div(long *bcd_low, long *bcd_high, long divisor);
long bcd2_mod(long *bcd_low, long *bcd_high, long modulo);
long bcd2_cmp(long *bcd_low, long *bcd_high, long compare);
```

bm_utils.c

```
#ifdef RCSID
static char *RCSid =
  "$Header: bm_utils.c 14-may-2001.20:59:33 mpoess Exp $ ";
#endif /* RCSID */

/* Copyright (c) 1999, 2001, Oracle Corporation.  All rights reserved.
*/

/*
  NAME
    bm_utils.c - <one-line expansion of the name>

  DESCRIPTION
    <short description of component this file declares/defines>

  PUBLIC FUNCTION(S)
    <list of external functions declared/defined - with one-line
descriptions>

  PRIVATE FUNCTION(S)
    <list of static functions defined in .c file - with one-line
descriptions>

  RETURNS
    <function return values, for .c file with single function>

  NOTES
    <other useful comments, qualifications, etc.>

  MODIFIED   (MM/DD/YY)
  mpoess      05/14/01 -
  mpoess      02/22/01 - add linux support
  mpoess      07/12/99 - exchange with new version July 8 1999
  mpoess      06/28/99 - Creation
  mpoess      06/28/99 - Creation

*/

/* @(#)bm_utils.c      2.1.6.1 */
/*
 *
 * Various routines that handle distributions, value selections and
 * seed value management for the DSS benchmark. Current functions:
 * env_config -- set config vars with optional environment override
 * yes_no -- ask simple yes/no question and return boolean result
 * a_rnd(min, max) -- random alphanumeric within length range
 * pick_str(size, set) -- select a string from the set of size
 * read_dist(file, name, distribution *) -- read named dist from file
 * tbl_open(path, mode) -- std fopen with lifenoise
 * julian(date) -- julian date correction
 * rowcnt(tbl) -- proper scaling of given table
 * e_str(set, min, max) -- build an embedded str
 * agg_str() -- build a string from the named set
 * dsscasecmp() -- version of strcasecmp()
 * dssncasecmp() -- version of strncasecmp()
 * getopt()
 * set_state() -- initialize the RNG
 */

#include "dss.h"
#include <stdio.h>
#include <time.h>
#include <errno.h>
#include <string.h>
#ifdef HP
#include <strings.h>
#endif          /* HP */
#include <ctype.h>
#include <math.h>
#ifndef _POSIX_SOURCE
#include <malloc.h>
#endif /* POSIX_SOURCE */
#include <fcntl.h>
#ifdef IBM
#include <sys/mode.h>
#endif /* IBM */
#include <sys/types.h>
#include <sys/stat.h>
/* Lines added by Chuck McDevitt for WIN32 support */
#ifdef WIN32
#ifndef _POSIX_
#include <io.h>
```

```
#ifndef S_ISREG
#define S_ISREG(m) ( ((m) & _S_IFMT) == _S_IFREG )
#define S_ISFIFO(m) ( ((m) & _S_IFMT) == _S_IFIFO )
#endif
#endif
#ifndef stat
#define stat _stat
#endif
#ifndef fdopen
#define fdopen _fdopen
#endif
#ifndef open
#define open _open
#endif
#ifndef O_RDONLY
#define O_RDONLY _O_RDONLY
#endif
#ifndef O_WRONLY
#define O_WRONLY _O_WRONLY
#endif
#ifndef O_CREAT
#define O_CREAT _O_CREAT
#endif
#endif
/* End of lines added by Chuck McDevitt for WIN32 support */
#include "dsstypes.h"


static char alpha_num[65] =
"0123456789abcdefghijklmnopqrstuvwxyz
ABCDEFGHIJKLMNOPQRSTUVWXYZ,";

#if defined(__STDC__) || defined(__cplusplus)
#define PROTO(s) s
#else
#define PROTO(s) ()
#endif

char    *getenv PROTO((const char *name));
void usage();
long *permute_dist(distribution *d, long stream);
extern long Seed[];

/*
 * env_config: look for a environmental variable setting and return its
 * value; otherwise return the default supplied
 */
char    *
env_config(char *var, char *dflt)
{
  static char *evar;

  if ((evar = getenv(var)) != NULL)
    return (evar);
  else
    return (dflt);
}

/*
 * return the answer to a yes/no question as a boolean
 */
long
yes_no(char *prompt)
{
   char     reply[128];

#ifdef WIN32
/* Disable warning about conditional expression is constant */
#pragma warning(disable:4127)
#endif
   while (1)
      {
#ifdef WIN32
#pragma warning(default:4127)
#endif
      printf("%s [Y/N]: ", prompt);
#ifdef LINUX
      fgets(reply,127,stdin);
#else
      gets(reply);
#endif
      switch (*reply)
         {
         case 'y':
         case 'Y':
            return (1);
         case 'n':
         case 'N':
            return (0);
         default:
            printf("Please answer 'yes' or 'no'.\n");
         }
      }
}

/*
 * generate a random string with length randomly selected in [min, max]
 * and using the characters in alphanum (currently includes a space
 * and comma)
 */
int
a_rnd(int min, int max, int column, char *dest)
{
  long     i,
           len,
           char_int;

  RANDOM(len, min, max, column);
  for (i = 0; i < len; i++)
    {
    if (i % 5 == 0)
     RANDOM(char_int, 0, MAX_LONG, column);
    *(dest + i) = alpha_num[char_int & 077];
    char_int >>= 6;
    }
  *(dest + len) = '\0';
  return (len);
}

/*
 * embed a randomly selected member of distribution d in alpha-
numeric
 * noise of a length rendomly selected between min and max at a
random
 * position
 */
void
e_str(distribution *d, int min, int max, int stream, char *dest)
{
   char strtmp[MAXAGG_LEN + 1];
   long loc;
   int len;

   a_rnd(min, max, stream, dest);
   pick_str(d, stream, strtmp);
   len = strlen(strtmp);
   RANDOM(loc, 0, (strlen(dest) - 1 - len), stream);
   strncpy(dest + loc, strtmp, len);

   return;
}
```

```
/*
 * return the string associate with the LSB of a uniformly selected
 * long in [1, max] where max is determined by the distribution
 * being queried
 */
int
pick_str(distribution *s, int c, char *target)
{
   long     i = 0;
   long     j;

   RANDOM(j, 1, s->list[s->count - 1].weight, c);
   while (s->list[i].weight < j)
      i++;
   strcpy(target, s->list[i].text);
   return(i);
}

/*
 * unjulian (long date) -- return(date - STARTDATE)
 */
long
unjulian(long date)
{
   int i;
   long res = 0;

   for (i = STARTDATE / 1000; i < date / 1000; i++)
      res += 365 + LEAP(i);
   res += date %  1000 - 1;

   return(res);
}

long
julian(long date)
{
   long      offset;
   long     result;
   long     yr;
   long     yend;

   offset = date - STARTDATE;
   result = STARTDATE;

#ifdef WIN32
/* Disable warning about conditional expression is constant */
#pragma warning(disable:4127)
#endif

   while (1)
      {
#ifdef WIN32
#pragma warning(default:4127)
#endif
      yr = result / 1000;
      yend = yr * 1000 + 365 + LEAP(yr);
      if (result + offset > yend)   /* overflow into next year */
         {
         offset -= yend - result + 1;
         result += 1000;
         continue;
         }
      else
         break;
      }
   return (result + offset);
}

/*
* load a distribution from a flat file into the target structure;
* should be rewritten to allow multiple dists in a file
*/
void
read_dist(char *path, char *name, distribution *target)
{
FILE     *fp;
char     line[256],
         token[256],
        *c;
long     weight,
         count = 0,
         name_set = 0;

   if (d_path == NULL)
                        {
                        sprintf(line, "%s%c%s",
                                 env_config(CONFIG_TAG,
CONFIG_DFLT), PATH_SEP, path);
                        fp = fopen(line, "r");
                        OPEN_CHECK(fp, line);
                        }
               else
                        {
                        fp = fopen(d_path, "r");
                        OPEN_CHECK(fp, d_path);
                        }
   while (fgets(line, sizeof(line), fp) != NULL)
      {
      if ((c = strchr(line, '\n')) != NULL)
         *c = '\0';
      if ((c = strchr(line, '#')) != NULL)
         *c = '\0';
      if (*line == '\0')
         continue;

      if (!name_set)
         {
         if (dsscasecmp(strtok(line, "\n\t "), "BEGIN"))
            continue;
         if (dsscasecmp(strtok(NULL, "\n\t "), name))
            continue;
         name_set = 1;
         continue;
         }
      else
         {
         if (!dssncasecmp(line, "END", 3))
            {
            fclose(fp);
            return;
            }
         }

      if (sscanf(line, "%[^|]|%ld", token, &weight) != 2)
         continue;

      if (!dsscasecmp(token, "count"))
         {
         target->count = weight;
         target->list =
            (set_member *)
               malloc((size_t)(weight * sizeof(set_member)));
         MALLOC_CHECK(target->list);
         target->max = 0;
         continue;
         }
      target->list[count].text =
         (char *) malloc((size_t)(strlen(token) + 1));
      MALLOC_CHECK(target->list[count].text);
      strcpy(target->list[count].text, token);
      target->max += weight;
      target->list[count].weight = target->max;
```

```
        count += 1;
        } /* while fgets() */

    if (count != target->count)
        {
        fprintf(stderr, "Read error on dist '%s'\n", name);
        fclose(fp);
        exit(1);
        }
                target->permute = (long *)NULL;
    fclose(fp);
    return;
}

/*
 * standard file open with life noise
 */

FILE    *
tbl_open(int tbl, char *mode)
{
    char      prompt[256];
    char      fullpath[256];
    FILE     *f;
    struct stat fstats;
    int       retcode;


    if (*tdefs[tbl].name == PATH_SEP)
        strcpy(fullpath, tdefs[tbl].name);
    else
        sprintf(fullpath, "%s%c%s",
            env_config(PATH_TAG, PATH_DFLT), PATH_SEP,
tdefs[tbl].name);

    retcode = stat(fullpath, &fstats);
    if (retcode && (errno != ENOENT))
        {
        fprintf(stderr, "stat(%s) failed.\n", fullpath);
        exit(-1);
        }
    if (S_ISREG(fstats.st_mode) && !force && *mode != 'r' )
        {
        sprintf(prompt, "Do you want to overwrite %s ?", fullpath);
        if (!yes_no(prompt))
            exit(0);
        }

    if (S_ISFIFO(fstats.st_mode))
        {
        retcode =
            open(fullpath, ((*mode ==
'r')?O_RDONLY:O_WRONLY)|O_CREAT);
        f = fdopen(retcode, mode);
        }
    else
        f = fopen(fullpath, mode);
    OPEN_CHECK(f, fullpath);
    if (header && columnar && tdefs[tbl].header != NULL)
        tdefs[tbl].header(f);

    return (f);
}


/*
 * agg_str(set, count) build an aggregated string from count unique
 * selections taken from set
 */
void
agg_str(distribution *set, long count, long col, char *dest)
{
                distribution *d;
                int i;

                d = set;
                *dest = '\0';

                for (i=0; i < count; i++)
                        {
                        strcat(dest,
DIST_MEMBER(set,*permute_dist(d, col)));
                        strcat(dest, " ");
                        d = (distribution *)NULL;
                        }
                *(dest + strlen(dest) - 1) = '\0';

    return;
}


long
dssncasecmp(char *s1, char *s2, int n)
{
    for (; n > 0; ++s1, ++s2, --n)
        if (tolower(*s1) != tolower(*s2))
            return ((tolower(*s1) < tolower(*s2)) ? -1 : 1);
        else if (*s1 == '\0')
            return (0);
        return (0);
}

long
dsscasecmp(char *s1, char *s2)
{
    for (; tolower(*s1) == tolower(*s2); ++s1, ++s2)
        if (*s1 == '\0')
            return (0);
    return ((tolower(*s1) < tolower(*s2)) ? -1 : 1);
}

#ifndef STDLIB_HAS_GETOPT
int optind = 0;
int opterr = 0;
char *optarg = NULL;

int
getopt(int ac, char **av, char *opt)
{
    static char *nextchar = NULL;
    char *cp;
    char hold;

    if (optarg == NULL)
        {
        optarg = (char *)malloc(BUFSIZ);
        MALLOC_CHECK(optarg);
        }

    if (!nextchar || *nextchar == '\0')
        {
        optind++;
        if (optind == ac)
            return(-1);
        nextchar = av[optind];
        if (*nextchar != '-')
            return(-1);
        nextchar +=1;
        }

    if (nextchar && *nextchar == '-')   /* -- termination */
        {
```

```
        optind++;
        return(-1);
        }
    else        /* found an option */
        {
        cp = strchr(opt, *nextchar);
        nextchar += 1;
        if (cp == NULL) /* not defined for this run */
            return('?');
        if (*(cp + 1) == ':')   /* option takes an argument */
            {
            if (*nextchar)
                {
                hold = *cp;
                cp = optarg;
                while (*nextchar)
                    *cp++ = *nextchar++;
                *cp = '\0';
                *cp = hold;
                }
            else        /* white space separated, use next arg */
                {
                if (++optind == ac)
                    return('?');
                strcpy(optarg, av[optind]);
                }
            nextchar = NULL;
            }
        return(*cp);
        }
}
#endif /* STDLIB_HAS_GETOPT */

char **
mk_ascdate(void)
{
    char **m;
    dss_time_t t;
    int i;

    m = (char**) malloc((size_t)(TOTDATE * sizeof (char *)));
    MALLOC_CHECK(m);
    for (i = 0; i < TOTDATE; i++)
        {
        m[i] = (char *)malloc(DATE_LEN * sizeof(char));
        MALLOC_CHECK(m[i]);
        mk_time((long)(i + 1), &t);
        strcpy(m[i], t.alpha);
        }

    return(m);
}

/*
 * set_state() -- initialize the RNG so that
 * appropriate data sets can be generated.
 * For each table that is to be generated, calculate the number of
rows/child, and send that to the
 * seed generation routine in speed_seed.c. Note: assumes that tables are
completely independent.
 * Returns the number of rows to be generated by the named step.
 */
long
set_state(int table, long sf, long procs, long step, long *extra_rows)
{
    int i;
            long rowcount, remainder, result;

    if (sf == 0 || step == 0)
        return(0);

            rowcount = tdefs[table].base / procs;
        printf ("procs %d MAX %d/n",procs,MAX_32B_SCALE);
            if ((sf / procs) > (int)MAX_32B_SCALE)
                    INTERNAL_ERROR("SCALE OVERFLOW.
RE-RUN WITH MORE CHILDREN.");
            rowcount *= sf;
            remainder = (tdefs[table].base % procs) * sf;
            rowcount += remainder / procs;
            result = rowcount;
            for (i=0; i < step - 1; i++)
                    {
                    if (table == LINE)      /* special case for shared
seeds */
                            tdefs[table].gen_seed(1, rowcount);
                    else
                            tdefs[table].gen_seed(0, rowcount);
                    /* need to set seeds of child in case there's a
dependency */
                    /* NOTE: this assumes that the parent and child
have the same base row count */
                            if (tdefs[table].child != NONE)

            tdefs[tdefs[table].child].gen_seed(0,rowcount);
                    }
            *extra_rows = remainder % procs;
            if (step > procs)       /* moving to the end to generate
updates */
                    tdefs[table].gen_seed(*extra_rows);

            return(result);
}
```

Build.c

```
#ifdef RCSID
static char *RCSid =
  "$Header: build.c 11-apr-2001.15:05:18 mpoess Exp $ ";
#endif /* RCSID */

/* Copyright (c) Oracle Corporation 1999, 2000. All Rights Reserved. */

/*

  NAME
    build.c - <one-line expansion of the name>

  DESCRIPTION
    <short description of component this file declares/defines>

  PUBLIC FUNCTION(S)
    <list of external functions declared/defined - with one-line
descriptions>

  PRIVATE FUNCTION(S)
    <list of static functions defined in .c file - with one-line
descriptions>

  RETURNS
    <function return values, for .c file with single function>

  NOTES
    <other useful comments, qualifications, etc.>

  MODIFIED   (MM/DD/YY)
  mpoess     04/11/01 -
  mpoess     08/25/00 - add support for SF=3000
  mpoess     07/12/99 - exchange with new version July 8 1999
  mpoess     06/28/99 - Creation
  mpoess     06/28/99 - Creation

*/


/* @(#)build.c          2.1.8.1 */
/* Sccsid:     @(#)build.c             9.1.1.17     11/15/95  12:52:28 */
```

```
/* stuff related to the customer table */
#include <stdio.h>
#include <string.h>
#ifndef VMS
#include <sys/types.h>
#endif
#if defined(SUN)
#include <unistd.h>
#endif
#include <math.h>

#include "dss.h"
#include "dsstypes.h"
#include "bcd2.h"
#ifdef ADHOC
#include "adhoc.h"
extern adhoc_t adhocs[];
#endif /* ADHOC */

#define LEAP_ADJ(yr, mnth)     \
((LEAP(yr) && (mnth) >= 2) ? 1 : 0)
#define JDAY_BASE       8035    /* start from 1/1/70 a la unix */
#define JMNTH_BASE      (-70 * 12) /* start from 1/1/70 a la unix */
#define JDAY(date) ((date) - STARTDATE + JDAY_BASE + 1)
#define PART_SUPP_BRIDGE(tgt, p, s) \
   { \
   long tot_scnt = tdefs[SUPP].base * scale; \
   tgt = (p + s *  (tot_scnt / SUPP_PER_PART +  \
            (long) ((p - 1) / tot_scnt))) % tot_scnt + 1; \
   }
#define RPRICE_BRIDGE(tgt, p) tgt = rpb_routine(p)
#define V_STR(avg, sd, tgt)  a_rnd((int)(avg * V_STR_LOW), \
(int)(avg * V_STR_HGH), sd, tgt)
#define TEXT(avg, sd, tgt)  \
dbg_text(tgt, (int)(avg * V_STR_LOW),(int)(avg * V_STR_HGH), sd)
static void gen_phone PROTO((long ind, char *target, long seed));

long
rpb_routine(long p)
            {
            long price;

            price = 90000;
            price += (p/10) % 20001;        /* limit contribution to $200
*/
            price += (p % 1000) * 100;

            return(price);
            }

static void
gen_phone(long ind, char *target, long seed)
            {
   long     acode,
                        exchg,
                        number;

   RANDOM(acode, 100, 999, seed);
   RANDOM(exchg, 100, 999, seed);
            RANDOM(number, 1000, 9999, seed);

            sprintf(target, "%02d", 10 + (ind % NATIONS_MAX));
   sprintf(target + 3, "%03d", acode);
   sprintf(target + 7, "%03d", exchg);
   sprintf(target + 11, "%04d", number);
   target[2] = target[6] = target[10] = '-';

   return;
            }

long
mk_cust(long n_cust, customer_t *c)
            {
            long i;

            c->custkey = n_cust;
            sprintf(c->name, C_NAME_FMT, C_NAME_TAG, n_cust);
            c->alen = V_STR(C_ADDR_LEN, C_ADDR_SD, c-
>address);
            RANDOM(i, 0, (nations.count - 1), C_NTRG_SD);
            c->nation_code = i;
            gen_phone(i, c->phone, (long)C_PHNE_SD);
            RANDOM(c->acctbal, C_ABAL_MIN, C_ABAL_MAX,
C_ABAL_SD);
            pick_str(&c_mseg_set, C_MSEG_SD, c->mktsegment);
            c->clen = TEXT(C_CMNT_LEN, C_CMNT_SD, c-
>comment);

            return (0);
            }



            /*
            * generate the numbered order and its associated lineitems
*/
void
mk_sparse (DSS_HUGE i, DSS_HUGE *ok, long seq)
            {
#ifndef SUPPORT_64BITS
            if (scale < MAX_32B_SCALE)
#endif
                        ez_sparse(i, ok, seq);
#ifndef SUPPORT_64BITS
            else
                        hd_sparse(i, ok, seq);
#endif
            return;
            }

            /*
            * the "simple" version of mk_sparse, used on systems with
64b support
            * and on all systems at SF <= 300G where 32b support is
sufficient
*/
void
ez_sparse(DSS_HUGE i, DSS_HUGE *ok, long seq)
            {
            long low_bits;
            LONG2HUGE(i, ok);
            low_bits = (long)(i & ((1 << SPARSE_KEEP) - 1));
            *ok = *ok >> SPARSE_KEEP;
            *ok = *ok << SPARSE_BITS;
            *ok += seq;
            *ok = *ok << SPARSE_KEEP;
            *ok += low_bits;



            return;
            }

#ifndef SUPPORT_64BITS
void
hd_sparse(long i, DSS_HUGE *ok, long seq)
            {
            long low_mask, seq_mask;
            static int init = 0;
            static DSS_HUGE *base, *res;

            if (init == 0)
```



```
                        {
                        INIT_HUGE(base);
                        INIT_HUGE(res);
                        init = 1;
                        }

                low_mask = (1 << SPARSE_KEEP) - 1;
                seq_mask = (1 << SPARSE_BITS) - 1;
                bin_bcd2(i, base, base + 1);
                HUGE_SET (base, res);
                HUGE_DIV (res, 1 << SPARSE_KEEP);
                HUGE_MUL (res, 1 << SPARSE_BITS);
                HUGE_ADD (res, seq, res);
                HUGE_MUL (res, 1 << SPARSE_KEEP);
                HUGE_ADD (res, *base & low_mask, res);
                bcd2_bin (&low_mask, *res);
                bcd2_bin (&seq_mask, *(res + 1));
                *ok = low_mask;
                *(ok + 1) = seq_mask;
                return;
                }
#endif

long
mk_order(DSS_HUGE index, order_t *o, long upd_num)
                {
                long    lcnt;
                long    rprice;
                long    ocnt;
                long    tmp_date;
                long    s_date;
                long    r_date;
                long    c_date;
                long    clk_num;
                long    supp_num;
                static char **asc_date = NULL;
                char tmp_str[2];
                char **mk_ascdate PROTO((void));
                int delta = 1;

                if (asc_date == NULL)
        asc_date = mk_ascdate();
                mk_sparse (index, o->okey,
                                (upd_num == 0) ? 0 : 1 + upd_num / (10000 /
refresh));
                RANDOM(o->custkey, O_CKEY_MIN, O_CKEY_MAX,
O_CKEY_SD);
   while (o->custkey % CUST_MORTALITY == 0)
                                {
                                o->custkey += delta;
                                o->custkey = MIN(o->custkey, O_CKEY_MAX);
                                delta *= -1;
                                }


   RANDOM(tmp_date, O_ODATE_MIN, O_ODATE_MAX,
O_ODATE_SD);
   strcpy(o->odate, asc_date[tmp_date - STARTDATE]);

   pick_str(&o_priority_set, O_PRIO_SD, o->opriority);
                RANDOM(clk_num, 1, MAX((scale * O_CLRK_SCL),
O_CLRK_SCL), O_CLRK_SD);
   sprintf(o->clerk, O_CLRK_FMT,
      O_CLRK_TAG,
      clk_num);
   o->clen = TEXT(O_CMNT_LEN, O_CMNT_SD, o->comment);
#ifdef DEBUG
                if (o->clen > O_CMNT_MAX) fprintf(stderr, "comment
error: O%d\n", index);
#endif /* DEBUG */
   o->spriority = 0;

   o->totalprice = 0;
   o->orderstatus = 'O';
   ocnt = 0;

                RANDOM(o->lines, O_LCNT_MIN, O_LCNT_MAX,
O_LCNT_SD);
   for (lcnt = 0; lcnt < o->lines; lcnt++)
                                {
      HUGE_SET(o->okey, o->l[lcnt].okey);
                        o->l[lcnt].lcnt = lcnt + 1;
                                RANDOM(o->l[lcnt].quantity, L_QTY_MIN,
L_QTY_MAX, L_QTY_SD);
                                RANDOM(o->l[lcnt].discount, L_DCNT_MIN,
L_DCNT_MAX, L_DCNT_SD);
      RANDOM(o->l[lcnt].tax, L_TAX_MIN, L_TAX_MAX,
L_TAX_SD);
      pick_str(&l_instruct_set, L_SHIP_SD, o->l[lcnt].shipinstruct);
      pick_str(&l_smode_set, L_SMODE_SD, o->l[lcnt].shipmode);
      o->l[lcnt].clen = TEXT(L_CMNT_LEN, L_CMNT_SD, o-
>l[lcnt].comment);
      RANDOM(o->l[lcnt].partkey, L_PKEY_MIN, L_PKEY_MAX,
L_PKEY_SD);
      RPRICE_BRIDGE( rprice, o->l[lcnt].partkey);
      RANDOM(supp_num, 0, 3, L_SKEY_SD);
      PART_SUPP_BRIDGE( o->l[lcnt].suppkey, o->l[lcnt].partkey,
supp_num);
      o->l[lcnt].eprice = rprice * o->l[lcnt].quantity;

      o->totalprice +=
         ((o->l[lcnt].eprice *
         ((long)100 - o->l[lcnt].discount)) / (long)PENNIES ) *
         ((long)100 + o->l[lcnt].tax)
         / (long)PENNIES;

                                RANDOM(s_date, L_SDTE_MIN,
L_SDTE_MAX, L_SDTE_SD);
      s_date += tmp_date;
                                RANDOM(c_date, L_CDTE_MIN,
L_CDTE_MAX, L_CDTE_SD);
      c_date += tmp_date;
                                RANDOM(r_date, L_RDTE_MIN,
L_RDTE_MAX, L_RDTE_SD);
      r_date += s_date;


      strcpy(o->l[lcnt].sdate, asc_date[s_date - STARTDATE]);
      strcpy(o->l[lcnt].cdate, asc_date[c_date - STARTDATE]);
      strcpy(o->l[lcnt].rdate, asc_date[r_date - STARTDATE]);


      if (julian(r_date) <= CURRENTDATE)
                                        {
         pick_str(&l_rflag_set, L_RFLG_SD, tmp_str);
         o->l[lcnt].rflag[0] = *tmp_str;
                                        }
      else
         o->l[lcnt].rflag[0] = 'N';

      if (julian(s_date) <= CURRENTDATE)
                                        {
         ocnt++;
         o->l[lcnt].lstatus[0] = 'F';
                                        }
      else
         o->l[lcnt].lstatus[0] = 'O';
                                }

   if (ocnt > 0)
      o->orderstatus = 'P';
   if (ocnt == o->lines)
      o->orderstatus = 'F';

                return (0);
}
```

```
long
mk_part(long index, part_t *p)
        {
        long    temp;
        long    snum;
        long    brnd;

        p->partkey = index;
        agg_str(&colors, (long)P_NAME_SCL, (long)P_NAME_SD,
p->name);
        RANDOM(temp, P_MFG_MIN, P_MFG_MAX,
P_MFG_SD);
        sprintf(p->mfgr, P_MFG_FMT, P_MFG_TAG, temp);
        RANDOM(brnd, P_BRND_MIN, P_BRND_MAX,
P_BRND_SD);
        sprintf(p->brand, P_BRND_FMT,
                P_BRND_TAG,
                (temp * 10 + brnd));
        p->tlen = pick_str(&p_types_set, P_TYPE_SD, p->type);
        p->tlen = strlen(p_types_set.list[p->tlen].text);
        RANDOM(p->size, P_SIZE_MIN, P_SIZE_MAX,
P_SIZE_SD);
        pick_str(&p_cntr_set, P_CNTR_SD, p->container);
        RPRICE_BRIDGE( p->retailprice, index);
        p->clen = TEXT(P_CMNT_LEN, P_CMNT_SD, p-
>comment);

        for (snum = 0; snum < SUPP_PER_PART; snum++)
                {
                p->s[snum].partkey = p->partkey;
                PART_SUPP_BRIDGE( p->s[snum].suppkey,
index, snum);
                RANDOM(p->s[snum].qty, PS_QTY_MIN,
PS_QTY_MAX, PS_QTY_SD);
                RANDOM(p->s[snum].scost, PS_SCST_MIN,
PS_SCST_MAX, PS_SCST_SD);
                p->s[snum].clen = TEXT(PS_CMNT_LEN,
PS_CMNT_SD, p->s[snum].comment);
                }
        return (0);
        }

long
mk_supp(long index, supplier_t *s)
        {
        long    i,
                bad_press,
                noise,
                offset,
                type;

        s->suppkey = index;
        sprintf(s->name, S_NAME_FMT, S_NAME_TAG, index);
        s->alen = V_STR(S_ADDR_LEN, S_ADDR_SD, s-
>address);
        RANDOM(i, 0, nations.count - 1, S_NTRG_SD);
        s->nation_code= i;
        gen_phone(i, s->phone, S_PHNE_SD);
        RANDOM(s->acctbal, S_ABAL_MIN, S_ABAL_MAX,
S_ABAL_SD);

        s->clen = TEXT(S_CMNT_LEN, S_CMNT_SD, s-
>comment);
        /* these calls should really move inside the if stmt below,
        * but this will simplify seedless parallel load
        */
        RANDOM(bad_press, 1, 10000, BBB_CMNT_SD);
        RANDOM(type, 0, 100, BBB_TYPE_SD);
        RANDOM(noise, 0, (s->clen - BBB_CMNT_LEN),
BBB_JNK_SD);
        RANDOM(offset, 0, (s->clen - (BBB_CMNT_LEN +
noise)),
                BBB_OFFSET_SD);
        if (bad_press <= S_CMNT_BBB)
                {
                type = (type < BBB_DEADBEATS) ?0:1;
    memcpy(s->comment + offset, BBB_BASE, BBB_BASE_LEN);
    if (type == 0)
                        memcpy(s->comment +
BBB_BASE_LEN + offset + noise,
                        BBB_COMPLAIN,
BBB_TYPE_LEN);
    else
                        memcpy(s->comment +
BBB_BASE_LEN + offset + noise,
                        BBB_COMMEND,
BBB_TYPE_LEN);
                }

        return (0);
        }

struct
        {
        char    *mdes;
        long    days;
        long    dcnt;
        }       months[] =

        {
                {NULL, 0, 0},
                {"JAN", 31, 31},
                {"FEB", 28, 59},
                {"MAR", 31, 90},
                {"APR", 30, 120},
                {"MAY", 31, 151},
                {"JUN", 30, 181},
                {"JUL", 31, 212},
                {"AUG", 31, 243},
                {"SEP", 30, 273},
                {"OCT", 31, 304},
                {"NOV", 30, 334},
                {"DEC", 31, 365}
                };

        long
                mk_time(long index, dss_time_t *t)
                {
                long    m = 0;
                long    y;
                long    d;

                t->timekey = index + JDAY_BASE;
                y = julian(index + STARTDATE - 1) / 1000;
                d = julian(index + STARTDATE - 1) % 1000;
                while (d > months[m].dcnt + LEAP_ADJ(y, m))
                        m++;
                PR_DATE(t->alpha, y, m,
                        d - months[m - 1].dcnt - ((LEAP(y)
&& m > 2) ? 1 : 0));
                t->year = 1900 + y;
                t->month = m + 12 * y + JMNTH_BASE;
                t->week = (d + T_START_DAY - 1) / 7 + 1;
                t->day = d - months[m - 1].dcnt - LEAP_ADJ(y,
m-1);

                return (0);
                }

        int
                mk_nation(long index, code_t *c)
```

```
                        {
                        c->code = index - 1;
                        c->text = nations.list[index - 1].text;
                        c->join = nations.list[index - 1].weight;
                        c->clen = TEXT(N_CMNT_LEN, N_CMNT_SD,
c->comment);
                        return(0);
                        }

                int
                        mk_region(long index, code_t *c)
                        {

                        c->code = index - 1;
                        c->text = regions.list[index - 1].text;
                        c->join = 0;        /* for completeness */
                        c->clen = TEXT(R_CMNT_LEN, R_CMNT_SD,
c->comment);
                        return(0);
                        }
```

config.h

```
/*
 * $Header: config.h 11-apr-2001.15:04:25 mpoess Exp $
 */

/* Copyright (c) Oracle Corporation 1999, 2000. All Rights Reserved. */

/* NOTE:  See 'header_template.doc' in the 'doc' dve under the 'forms'
     directory for the header file template that includes instructions.
*/

/*
  NAME
    <<<LOWERCASE_FILENAME>>>.h - <one-line expansion of the
name>

  DESCRIPTION
    <short description of component this file declares/defines>

  RELATED DOCUMENTS

  INSPECTION STATUS
    Inspection date:
    Inspection status:
    Estimated increasing cost defects per page:
    Rule sets:

  ACCEPTANCE REVIEW STATUS
    Review date:
    Review status:
    Reviewers:

  PUBLIC FUNCTION(S)
    <list of external functions declared/defined - with one-line
descriptions>

  PRIVATE FUNCTION(S)
    <list of static functions defined in .c file - with one-line
descriptions>

  EXAMPLES

  NOTES
    <other useful comments, qualifications, etc.>

  MODIFIED   (MM/DD/YY)
  mpoess     04/11/01 -
  mpoess     08/28/00 - add support for SF > 1000
  mpoess     07/12/99 - exchange with new version July 8 1999
  mpoess     06/28/99 - Creation
  mpoess     06/28/99 - Creation
*/
/*
 * Sccsid:     @(#)config.h           2.1.8.2
 *
 * this file allows the compilation of DBGEN to be tailored to specific
 * architectures and operating systems. Some options are grouped
 * together to allow easier compilation on a given vendor's hardware.
 *
 * The following #defines will effect the code:
 *   TPCH             -- make will create TPCH (set in makefile)
 *   TPCR             -- make will create TPCR (set in makefile)
 *   KILL(pid)        -- how to terminate a process in a parallel load
 *   SPAWN            -- name of system call to clone an existing process
 *   SET_HANDLER(proc) -- name of routine to handle signals in
parallel load
 *   WAIT(res, pid)    -- how to await the termination of a child
 *   SEPARATOR        -- character used to separate fields in flat files
 *   DBNAME           -- default name of database to be loaded
 *   STDLIB_HAS_GETOPT -- to prevent confilcts with gloabal
getopt()
 *   MDY_DATE         -- generate dates as MM-DD-YY
 *   WIN32            -- support for WindowsNT
 *   SUPPORT_64BITS    -- compiler defines a 64 bit datatype
 *   DSS_HUGE         -- 64 bit data type
 *   HUGE_FORMAT       -- printf string for 64 bit data type
 *   HUGE_COUNT        -- number of objects in DSS_HUGE
 *   EOL_HANDLING      -- flat files don't need final column separator
 *
 *   OS defines
 *   ==========
 *   ATT       -- getopt() handling
 *   DOS       -- disable all multi-user functionality/dependency
 *   HP        -- posix source inclusion differences
 *   IBM       -- posix source inclusion differences
 *   ICL       -- getopt() handling
 *   MVS       -- special handling of varchar format
 *   SGI       -- getopt() handling
 *   SUN       -- getopt() handling
 *   TANDEM     -- EOL handling
 *   U2200      -- death of parent kills children automatically
 *   VMS       -- signal/fork handing differences
 *
 *   Database defines
 *   ================
 *   DB2       -- use DB2 dialect in QGEN
 *   INFORMIX   -- use Informix dialect in QGEN
 *   SQLSERVER  -- use SQLSERVER dialect in QGEN
 *   SYBASE     -- use Sybase dialect in QGEN
 *   TDAT       -- use Teradata dialect in QGEN
 */

#ifdef DOS
#define DSS_PROC       1
#define PATH_SEP   '\\'
#else


#ifdef ATT
#define STDLIB_HAS_GETOPT
#ifdef SQLSERVER
#define WIN32
#else
/* the 64 bit defines are for the Metaware compiler */
#define SUPPORT_64BITS
#define DSS_HUGE long long
#define HUGE_COUNT            1
#define HUGE_FORMAT "%LLd"
#endif /* SQLSERVER or MP/RAS */
#endif /* ATT */

#ifdef HP
#define _INCLUDE_POSIX_SOURCE
#define STDLIB_HAS_GETOPT
```

```
#endif /* HP */

#ifdef IBM
#define _POSIX_SOURCE
/*
 * if the C compiler is 3.1 or later, then uncomment the
 * lines for 64 bit seed generation
 */
/* #define SUPPORT_64BITS */
/* #define DSS_HUGE long long */
/* #define HUGE_COUNT        1 */
#define STDLIB_HAS_GETOPT
#endif /* IBM */

#ifdef ICL
#define STDLIB_HAS_GETOPT
#endif /* ICL */
#define SUN
#ifdef SUN
#define STDLIB_HAS_GETOPT
#define SUPPORT_64BITS
#define DSS_HUGE long long
#define HUGE_FORMAT "%lld"
#define HUGE_COUNT            1
#endif /* SUN */

#ifdef SGI
#define STDLIB_HAS_GETOPT
#define SUPPORT_64BITS
#define DSS_HUGE __uint64_t
#define HUGE_COUNT            1
#endif /* SGI */

#ifdef TANDEM
#define EOL_HANDLING
#endif /* TANDEM */

#ifdef VMS
#define SPAWN   vfork
#define KILL(pid) kill(SIGQUIT, pid)
#define SET_HANDLER(proc) signal(SIGQUIT, proc)
#define WAIT(res, pid) wait(res)
#define SIGS_DEFINED
#endif /* VMS */

#if (defined(WIN32)&&!defined(_POSIX_))
#define pid_t int
#define SET_HANDLER(proc) signal(SIGINT, proc)
#define KILL(pid) \

TerminateProcess(OpenProcess(PROCESS_TERMINATE,FALSE,pid),
3)
#if (defined (__WATCOMC__))
#define SPAWN()   spawnv(P_NOWAIT, spawn_args[0], spawn_args)
#define WAIT(res, pid) cwait(res, pid, WAIT_CHILD)
#else
#define SPAWN()   _spawnv(_P_NOWAIT, spawn_args[0],
spawn_args)
#define WAIT(res, pid) _cwait(res, pid, _WAIT_CHILD)
#define getpid          _getpid
#endif /* WATCOMC */
#define SIGS_DEFINED
#define PATH_SEP    '\\'
#ifndef TEST_32B
#define SUPPORT_64BITS
#define DSS_HUGE __int64
#define HUGE_COUNT            1
#define HUGE_FORMAT "%I64d"
#endif /* TEST_32B */
/* need to define process termination codes to match UNIX */
/* these are copied from Linux/GNU and need to be verified as part of a
rework of */
/* process handling under NT (29 Apr 98) */
#define WIFEXITED(s)            ((s & 0xFF) == 0)
#define WIFSIGNALED(s)          (((unsigned int)((status)-1) & 0xFFFF)
< 0xFF)
#define WIFSTOPPED(s)           (((s) & 0xff) == 0x7f)
#define WTERMSIG(s)                     ((s) & 0x7f)
#define WSTOPSIG(s)                     (((s) & 0xff00) >> 8)
#endif /* WIN32 */

#ifndef SIGS_DEFINED
#define KILL(pid) kill(SIGUSR1, pid)
#define SET_HANDLER(proc) signal(SIGUSR1, proc)
#define SPAWN   fork
#define WAIT(res, pid) wait(res)
#endif /* DEFAULT */

#define DSS_PROC        getpid()
#endif /* DOS */

#ifndef DBNAME
#define DBNAME "dss"
#endif /* DBNAME */

#ifndef PATH_SEP
#define PATH_SEP '/'
#endif /* PATH_SEP */

#ifndef DSS_HUGE
#define DSS_HUGE   long
#define HUGE_COUNT            2
#endif
```

driver.c

```
#ifdef RCSID
static char *RCSid =
  "$Header: driver.c 11-apr-2001.14:36:31 mpoess Exp $ ";
#endif /* RCSID */

/* Copyright (c) Oracle Corporation 1999, 2000. All Rights Reserved. */

/*

  NAME
    driver.c - <one-line expansion of the name>

  DESCRIPTION
    <short description of component this file declares/defines>

  PUBLIC FUNCTION(S)
    <list of external functions declared/defined - with one-line
descriptions>

  PRIVATE FUNCTION(S)
    <list of static functions defined in .c file - with one-line
descriptions>

  RETURNS
    <function return values, for .c file with single function>

  NOTES
    <other useful comments, qualifications, etc.>

  MODIFIED   (MM/DD/YY)
  mpoess     04/11/01 -
  mpoess     08/25/00 - add support for SF=3000
  mpoess     08/31/99 - add Jacks changes for version 1.1.0/1.2.0 (H/R)
as of
  mpoess     07/12/99 - exchange with new version July 8 1999
```

```
   mpoess     06/28/99 - Creation
   mpoess     06/28/99 - Creation

*/

/* @(#)driver.c          2.1.8.4 */
/* main driver for dss banchmark */

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

#include "config.h"
#include <stdlib.h>
#if (defined(_POSIX_)||!defined(WIN32))            /* Change for
Windows NT */
#include <unistd.h>
#include <sys/wait.h>
#endif /* WIN32 */
#include <stdio.h>                              /* */
#include <limits.h>
#include <math.h>
#include <ctype.h>
#include <signal.h>
#include <string.h>
#include <errno.h>
#ifdef HP
#include <strings.h>
#endif
#if (defined(WIN32)&&!defined(_POSIX_))
#include <process.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 NOSYSMETRICS
#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 "dsstypes.h"
#include "bcd2.h"

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

#ifdef ORACLE
#include "part.h"
int partition = 0;      /* partition flag       */
vals *values;
FILE *in;               /* fdes for input file  */
FILE  **out;              /* output fd   array    */
FILE *par;              /* fdes for params file */
#ifdef DEBUG
/* for debugging */
FILE *debug_fd = NULL;  /* fd for debugging file */
char debug_fn[FLEN];    /* debug file name       */
extern long row_count;
extern long *part_row;
int j;
#endif /* DEBUG */
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 /* ORACLE */

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       (*verify) ();    -- function to verfiy the data set without building
it
* int       child;          -- non-zero if there is an associated detail table
* unsigned long vtotal;      -- "checksum" total
* }         tdef;
*
*/

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

```
int pr_nation (code_t * c, int mode);
int pr_region (code_t * c, int mode);

/*
* inline load functions; used with -D(irect) option
*/
int ld_cust (customer_t * c, int mode);
int ld_line (order_t * o, int mode);
int ld_order (order_t * o, int mode);
int ld_part (part_t * p, int mode);
int ld_psupp (part_t * p, int mode);
int ld_supp (supplier_t * s, int mode);
int ld_order_line (order_t * o, int mode);
int ld_part_psupp (part_t * p, int mode);
int ld_nation (code_t * c, int mode);
int ld_region (code_t * c, int mode);

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

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

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


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

int *pids;


/*
* 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 < 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);
                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);

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

/*
* generate a particular table
*/
void
gen_tbl (int tnum, DSS_HUGE start, long count, long upd_num)
{
        static order_t o;
        supplier_t supp;
        customer_t cust;
        part_t part;
        code_t code;
        static int completed = 0;
        static int init = 0;
        DSS_HUGE i;

        int rows_per_segment=0;
        int rows_this_segment=-1;
        int residual_rows=0;

        if (insert_segments)
          {
                rows_per_segment = count / insert_segments;
                residual_rows = count - (rows_per_segment *
insert_segments);
          }
        if (init == 0)
        {
                INIT_HUGE(o.okey);
                for (i=0; i < O_LCNT_MAX; i++)
                        INIT_HUGE(o.l[i].okey);
                init = 1;
        }

        for (i = start; count; count--, i++)
        {
                LIFENOISE (1000, i);
                row_start(tnum);

                switch (tnum)
                {
                case LINE:
                case ORDER:
                case ORDER_LINE:
                     mk_order (i, &o, upd_num % 10000);

                  if (insert_segments  && (upd_num > 0))
                        if((upd_num / 10000) <
residual_rows)
                                {
                                if((++rows_this_segment)
> rows_per_segment)
                                        {


        rows_this_segment=0;
                                        upd_num +=
10000;
                                        }
                                }
                        else
                                {
                                if((++rows_this_segment)
>= rows_per_segment)
                                        {


        rows_this_segment=0;
                                        upd_num +=
10000;
                                        }
                                }
```

```
                        if (set_seeds == 0)
                                if (validate)

        tdefs[tnum].verify(&o, 0);
                                else

        tdefs[tnum].loader[direct] (&o, upd_num);
                        break;
                case SUPP:
                        mk_supp (i, &supp);
                        if (set_seeds == 0)
                                if (validate)

        tdefs[tnum].verify(&supp, 0);
                                else

        tdefs[tnum].loader[direct] (&supp, upd_num);
                        break;
                case CUST:
                        mk_cust (i, &cust);
                        if (set_seeds == 0)
                                if (validate)

        tdefs[tnum].verify(&cust, 0);
                                else

        tdefs[tnum].loader[direct] (&cust, upd_num);
                        break;
                case PSUPP:
                case PART:
                case PART_PSUPP:
                        mk_part (i, &part);
                        if (set_seeds == 0)
                                if (validate)

        tdefs[tnum].verify(&part, 0);
                                else

        tdefs[tnum].loader[direct] (&part, upd_num);
                        break;
                case NATION:
                        mk_nation (i, &code);
                        if (set_seeds == 0)
                                if (validate)

        tdefs[tnum].verify(&code, 0);
                                else

        tdefs[tnum].loader[direct] (&code, 0);
                        break;
                case REGION:
                        mk_region (i, &code);
                        if (set_seeds == 0)
                                if (validate)

        tdefs[tnum].verify(&code, 0);
                                else

        tdefs[tnum].loader[direct] (&code, 0);
                        break;
                }
                row_stop(tnum);
                if (set_seeds && (i % tdefs[tnum].base) < 2)
                {
                        printf("\nSeeds for %s at rowcount
%ld\n", tdefs[tnum].comment, i);
                        dump_seeds(tnum);
                }
        }
        completed |= 1 << tnum;
}



void
usage (void)
{
        fprintf (stderr, "%s\n%s\n\t%s\n%s %s\n\n",
                        "USAGE:",
                        "dbgen [-{vfFD}] [-O {fhmsv}][-T
{pcsoPSOL}]",
                        "[-s <scale>][-C <procs>][-S <step>]",
                        "dbgen [-v] [-O {dfhmr}] [-s <scale>]",
                        "[-U <updates>] [-r <percent>]");
        fprintf (stderr, "-b <s> -- load distributions for <s>\n");
        fprintf (stderr, "-C <n> -- use <n> processes to generate
data\n");
        fprintf (stderr, "          [Under DOS, must be used with -
S]\n");
        fprintf (stderr, "-D     -- do database load in line\n");
    fprintf (stderr, "-d <n> -- split deletes between <n> files\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, "-i <n> -- split inserts between <n> files\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 v   -- Verify data set without generating
it.\n");
        fprintf (stderr, "-q     -- enable QUIET mode\n");
        fprintf (stderr, "-r <n> -- updates refresh (n/100)%% of
the\n");
        fprintf (stderr, "          data set\n");
        fprintf (stderr, "-s <n> -- set Scale Factor (SF) to  <n> \n");
        fprintf (stderr, "-S <n> -- build the <n>th step of the
data/update set\n");
        fprintf (stderr, "-T c   -- generate cutomers 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/lineitem 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 ORACLE
    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, "<column type>\n");
    fprintf(stderr, "<column pos> - 1 for first column, 2 for 2nd column,
...\n");
    fprintf(stderr, "<partition val1>\n");
    fprintf(stderr, "<partition val2>\n");
    fprintf(stderr, "...\n");
```

```
    fprintf(stderr, "<partition valm>\n");
    fprintf(stderr, "Notes:\n");
    fprintf(stderr, "%s\n",
          "Column types are: 0 for numbers, 1 for strings, 2 for dates");
    fprintf(stderr, "==================================\n\n");
#endif /* ORACLE */
            fprintf (stderr,
                    "\nTo generate the SF=1 (1GB), validation
database population, use:\n");
            fprintf (stderr, "\tdbgen -vfF -s 1\n");
            fprintf (stderr, "\nTo generate updates for a SF=1 (1GB),
use:\n");
            fprintf (stderr, "\tdbgen -v -U 1 -s 1\n");
}

/*
* pload() -- handle the parallel loading of tables
*/
#ifndef DOS
/*
* int partial(int tbl, int s) -- generate the s-th part of the named tables
data
*/
int
partial (int tbl, int s)
{
            long long rowcnt;
            long extra;

            if (verbose > 0)
            {
                    fprintf (stderr, "\tStarting to load stage %d of %d
for %s...",
                            s, children, tdefs[tbl].comment);
            }

            if (direct == 0)
                    set_files (tbl, s);

            rowcnt = (long long)set_state(tbl, scale, children, s, &extra);

            if (s == children)
                    gen_tbl (tbl, rowcnt * (s - 1) + 1, rowcnt + extra,
upd_num);
            else
                    gen_tbl (tbl, rowcnt * (s - 1) + 1, rowcnt,
upd_num);

            if (verbose > 0)
                    fprintf (stderr, "done.\n");

            return (0);
}


int
pload (int tbl)
{
            int c = 0, i, status;

            if (verbose > 0)
            {
                    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+1);
                            exit (0);
                    }
                    else if (verbose > 0)                   /*
PARENT */
                            fprintf (stderr, ".");
            }

            if (verbose > 0)
                    fprintf (stderr, "waiting...");

            c = children;
            while (c)
            {
                    i = WAIT (&status, pids[c - 1]);
                    if (i == -1 && children)
                    {
                            if (errno == ECHILD)
                                    fprintf (stderr, "\nCould not
wait on pid %d\n", pids[c - 1]);
                            else if (errno == EINTR)
                                    fprintf (stderr, "\nProcess
%d stopped abnormally\n", pids[c - 1]);
                            else if (errno == EINVAL)
                                    fprintf (stderr, "\nProgram
bug\n");
                    }
                    if (! WIFEXITED(status)) {
                            (void) fprintf(stderr, "\nProcess %d: ",
i);
                            if (WIFSIGNALED(status)) {
                                    (void) fprintf(stderr, "rcvd
signal %d\n",

            WTERMSIG(status));
                                    } else if
(WIFSTOPPED(status)) {
                                    (void) fprintf(stderr,
"stopped, signal %d\n",

            WSTOPSIG(status));
                                                    }

                            }
                    c--;
            }

            if (verbose > 0)
                    fprintf (stderr, "done\n");
            return (0);
}
#endif /* !DOS */

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

#ifdef ORACLE
   /* set some default */
   if (partition) {
     (void) sprintf(param,"%s","./part.param");
     (void) sprintf(outname,"%s","/dev/sqlldr");
   }
#endif /* ORACLE */
#ifndef ORACLE
            while ((option = getopt (count, vector,
                    "b:C:Dd:Ffi:hn:O:P:qr:s:S:T:U:v")) != -1)
```

```
#else /* ORACLE */
        while ((option = getopt (count, vector,
                "b:C:Dd:Ffi:hn:O:P:qr:s:S:T:U:vI:o:a:p")) != -1)
#endif /* ORACLE */
        switch (option)
                {
#ifdef ORACLE
        case 'p':
          partition = 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 /* ORACLE */
                case 'b':                                       /*
load distributions from named file */
                        d_path = (char *)malloc(strlen(optarg)
+ 1);
                        MALLOC_CHECK(d_path);
                        strcpy(d_path, optarg);
                        break;
                case 'q':                                       /* all
prompts disabled */
                        verbose = -1;
                        break;
                case 'i':
                        insert_segments = atoi (optarg);
                        break;
                case 'd':
                        delete_segments = atoi (optarg);
                        break;
        case 'S':                               /* generate a
particular STEP */
          step = atoi (optarg);
          break;
        case 'v':                               /* life noises
enabled */
          verbose = 1;
          break;
        case 'f':                               /* blind
overwrites; Force */
          force = 1;
          break;
        case 'T':                               /* generate a
specifc 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;
          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 <
REGION; i++)
                        {
                                tdefs[i].base
*= flt_scale;
                                if (tdefs[i].base
< 1)

tdefs[i].base = 1;
                        }
                }
                else
                        scale = (long) flt_scale;
                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");
                                fprintf (stderr,
                                        "Your resulting
data set MAY NOT BE COMPLIANT!\n");
                        }
                        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':                       /*
calibrate the RNG usage */
                                set_seeds = 1;
                                break;
                        case 'v':                       /*
validate the data set */
                                validate = 1;
                                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;
#ifndef DOS
                        case 'C':
                                children = atoi (optarg);
                                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,
                                        "%s
Population Generator (Version %d.%d.%d%s)\n",
                                        NAME,
VERSION, RELEASE,

MODIFICATION, PATCH);
                                fprintf (stderr, "Copyright
%s %s\n", TPC, C_DATES);
                                usage ();
                                exit (1);
                }

#ifndef DOS
        if (children != 1 && step == -1)
                {
                pids = malloc(children * sizeof(pid_t));
                MALLOC_CHECK(pids)
                }
#else
        if (children != 1 && step < 0)
                {
                fprintf(stderr, "ERROR: -C must be accompanied
by -S on this platform\n");
                exit(1);
                }
#endif /* DOS */

        return;
}

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

        table = (1 << CUST) |
                (1 << SUPP) |
                (1 << NATION) |
                (1 << REGION) |
                (1 << PART_PSUPP) |
                (1 << ORDER_LINE);
        force = 0;
   insert_segments=0;
   delete_segments=0;
   insert_orders_segment=0;
   insert_lineitem_segment=0;
   delete_segment=0;
        verbose = 0;
        columnar = 0;
        set_seeds = 0;
        header = 0;
        direct = 0;
        scale = 1;
        flt_scale = 1.0;
        updates = 0;
```

```
        refresh = UPD_PCT;
        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;
        d_path = NULL;

#ifdef NO_SUPPORT
        signal (SIGINT, exit);
#endif /* NO_SUPPORT */
        process_options (ac, av);
#ifdef ORACLE
   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 /* ORACLE */
#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

        if (verbose >= 0)
                {
                fprintf (stderr,
                        "%s Population Generator (Version
%d.%d.%d%s)\n",
                        NAME, 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)
              {
                /*
                 * set RNG to start generating rows beyond
SF=scale
                 */
                double fix1;
            set_state (ORDER, scale, 1, 2, (long *)&i);
                fix1 = (double)tdefs[ORDER_LINE].base /
(double)10000;
            rowcnt = (int)(fix1 * scale * refresh);
                if (step > 0)
                        {
                        /*
                         * adjust RNG for any prior update
generation
                         */
                        sd_order(0, rowcnt * (step - 1));
                        sd_line(0, rowcnt * (step - 1));
                        upd_num = step - 1;
                        }
                else
                        upd_num = 0;

                while (upd_num < updates)
                        {
                        if (verbose > 0)
                                fprintf (stderr,
                                "Generating update pair
#%d for %s [pid: %d]",
                                upd_num + 1,
tdefs[ORDER_LINE].comment, DSS_PROC);
                        insert_orders_segment=0;
                        insert_lineitem_segment=0;
                        delete_segment=0;
                        minrow = upd_num * rowcnt + 1;
                        gen_tbl (ORDER_LINE, minrow,
rowcnt, upd_num + 1);
                        if (verbose > 0)
                                fprintf (stderr, "done.\n");
                        pr_drange (ORDER_LINE, minrow,
rowcnt, upd_num + 1);
                        upd_num++;
                        }

                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);
                                }
                }

/*
 * traverse the tables, invoking the appropriate data generation routine
for any to be built
 */
        for (i = PART; i <= REGION; i++)
                if (table & (1 << i))
                {
#ifdef ORACLE
#ifdef DEBUG
        row_count = 0;
        for (j=0; j<numpart; j++) {
```

```
          part_row[j] = 0;
        }
        sprintf(debug_fn,"%s.%s.%d.out", "dbgen", tdefs[i].name,
step+1);
        if ((debug_fd = fopen(debug_fn, "w+")) == NULL) {
          fprintf(stderr, "Problem opening %s\n", debug_fn);
          exit(-1);
        }
#endif /* DEBUG */
#endif /* ORACLE */
                                        if (children > 1 && i < NATION)
                                                if (step >= 0)
                                                {
                                                        if (validate)
                                                        {

        INTERNAL_ERROR("Cannot validate parallel data
generation");
                                                        }
                                                        else

        partial (i, step);
                                                }
#ifdef DOS
                                                else
                                                {
                                                        fprintf (stderr,

        "Parallel load is not supported on your platform.\n");
                                                        exit (1);
                                                }
#else
                                                else
                                                {
                                                        if (validate)
                                                        {

        INTERNAL_ERROR("Cannot validate parallel data
generation");
                                                        }
                                                        else
                                                                pload
(i);
                                                }
#endif /* DOS */
                                                else
                                                {
                                                        minrow = 1;
                                                        if (i < NATION)

        rowcnt = tdefs[i].base * scale;
                                                        else

        rowcnt = tdefs[i].base;
                                                        if (verbose > 0)

        fprintf (stderr, "%s data for %s [pid: %ld]",

        (validate)?"Validating":"Generating", tdefs[i].comment,
DSS_PROC);
                                                        gen_tbl (i,
minrow, rowcnt, upd_num);
                                                        if (verbose > 0)

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

        printf("Validation checksum for %s at %d GB: %0x\n",

tdefs[i].name, scale, tdefs[i].vtotal);
                        }
                        if (direct)
                                close_direct ();

                        return (0);
}
```

dss.h

```
/*
 * $Header: dss.h 11-apr-2001.15:00:22 mpoess Exp $
 */

/* Copyright (c) Oracle Corporation 1999, 2000. All Rights Reserved. */

/* NOTE:  See 'header_template.doc' in the 'doc' dve under the 'forms'
     directory for the header file template that includes instructions.
*/

/*
   NAME
     <<<LOWERCASE_FILENAME>>>.h - <one-line expansion of the
name>

   DESCRIPTION
     <short description of component this file declares/defines>

   RELATED DOCUMENTS

   INSPECTION STATUS
     Inspection date:
     Inspection status:
     Estimated increasing cost defects per page:
     Rule sets:

   ACCEPTANCE REVIEW STATUS
     Review date:
     Review status:
     Reviewers:

   PUBLIC FUNCTION(S)
     <list of external functions declared/defined - with one-line
descriptions>

   PRIVATE FUNCTION(S)
     <list of static functions defined in .c file - with one-line
descriptions>

   EXAMPLES

   NOTES
     <other useful comments, qualifications, etc.>

   MODIFIED   (MM/DD/YY)
   mpoess      04/11/01 -
   mpoess      02/22/01 - add linux support
   mpoess      08/29/00 - add support for SF>1000
   mpoess      08/31/99 - add Jacks changes for version 1.1.0/1.2.0 (H/R)
as of
   mpoess      07/12/99 - exchange with new version July 8 1999
   mpoess      06/28/99 - Creation
   mpoess      06/28/99 - Creation

*/
/*
 * Sccsid:     @(#)dss.h               2.1.8.5
 *
 * general definitions and control information for the DSS code
 * generator; if it controls the data set, it's here
 */
#ifndef DSS_H
#define  DSS_H
#ifdef TPCH
#define NAME                            "TPC-H"
```

```
#define VERSION         1
#define RELEASE         3
#define MODIFICATION    0
#define PATCH           ""
#endif
#ifdef TPCR
#define NAME                                    "TPC-R"
#define VERSION         1
#define RELEASE         3
#define MODIFICATION    0
#define PATCH           ""
#endif
#ifndef NAME
#error Benchmark version must be defined in config.h
#endif
#define TPC             "Transaction Processing Performance Council"
#define C_DATES         "1994 - 2000"

#include "config.h"
#include "shared.h"

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

#define  NONE                   -1
#define  PART                   0
#define  PSUPP                  1
#define  SUPP                   2
#define  CUST                   3
#define  ORDER                  4
#define  LINE                   5
#define  ORDER_LINE 6
#define  PART_PSUPP 7
#define  NATION                 8
#define  REGION                 9
#define  UPDATE                 10
#define  MAX_TABLE              11
#define  ONE_STREAM             1
#define  ADD_AT_END             2

#ifdef MAX
#undef MAX
#endif
#ifdef MIN
#undef MIN
#endif
#define MAX(a,b) ((a > b )?a:b)
#define MIN(A,B)  ( (A) < (B) ? (A) : (B))

#define INTERNAL_ERROR(p)  {fprintf(stderr,"%s", p);abort();}
#define LN_CNT  4
static char lnoise[4] = {'|', '/', '-', '\\' };
#define LIFENOISE(n, var)       \
                if (verbose > 0) fprintf(stderr, "%c\b",
lnoise[(var%LN_CNT)])

#define MALLOC_CHECK(var) \
   if ((var) == NULL) \
      { \
      fprintf(stderr, "Malloc failed at %s:%d\n",  \
         __FILE__, __LINE__); \
      exit(1);\
      }
#define OPEN_CHECK(var, path) \
   if ((var) == NULL) \
      { \
      fprintf(stderr, "Open failed for %s at %s:%d\n",  \
         path, __FILE__, __LINE__); \
      exit(1);\
      }
#ifndef MAX_CHILDREN
#define MAX_CHILDREN    1000
#endif

/*
 * macros that control sparse keys
 *
 * refer to Porting.Notes for a complete explanation
 */
#ifndef BITS_PER_LONG
#define BITS_PER_LONG   32
#define MAX_LONG        0x7FFFFFFF
#endif /* BITS_PER_LONG */
#define SPARSE_BITS     2
#define SPARSE_KEEP     3
#define  MK_SPARSE(key, seq) \
        (((((key>>3)<<2)|(seq & 0x0003))<<3)|(key & 0x0007))

#define RANDOM(tgt, lower, upper, stream)
                dss_random(&tgt, lower, upper, stream)



typedef struct
{
  long      weight;
  char     *text;
}         set_member;

typedef struct
{
  int       count;
  int       max;
  set_member *list;
  long *permute;
}         distribution;
/*
 * some handy access functions
 */
#define DIST_SIZE(d)                            d->count
#define DIST_MEMBER(d, i)     ((set_member *)((d)->list + i))->text

typedef struct
{
  char     *name;
  char     *comment;
  long      base;
  int       (*header) ();
  int       (*loader[2]) ();
  long      (*gen_seed)();
  int       (*verify) ();
  int       child;
  unsigned long vtotal;
}         tdef;

typedef struct SEED_T {
                long table;
                long value;
                long usage;
                long boundary;
                } seed_t;


#if defined(__STDC__)
#define PROTO(s) s
#else
#define PROTO(s) ()
#endif

/* bm_utils.c */
char          *env_config PROTO((char *var, char *dflt));
long          yes_no PROTO((char *prompt));
```

```
int     a_rnd PROTO((int min, int max, int column, char *dest));
int     tx_rnd PROTO((long min, long max, long column, char *tgt));
long        julian PROTO((long date));
long        unjulian PROTO((long date));
FILE        *tbl_open PROTO((int tbl, char *mode));
long        dssncasecmp PROTO((char *s1, char *s2, int n));
long        dsscasecmp PROTO((char *s1, char *s2));
int                 pick_str PROTO((distribution * s, int c, char
*target));
void        agg_str PROTO((distribution *set, long count, long col, char
*dest));
void        read_dist PROTO((char *path, char *name, distribution *
target));
void        embed_str PROTO((distribution *d, int min, int max, int
stream, char *dest));
#ifndef STDLIB_HAS_GETOPT
int                 getopt PROTO((int arg_cnt, char **arg_vect, char
*oprions));
#endif /* STDLIB_HAS_GETOPT */
long        set_state PROTO((int t, long scale, long procs, long step,
long *e));

/* rnd.c */
long        NextRand PROTO((long nSeed));
long        UnifInt PROTO((long nLow, long nHigh, long nStream));
double      UnifReal PROTO((double dLow, double dHigh, long
nStream));
double      Exponential PROTO((double dMean, long nStream));
void        dss_random(long *tgt, long min, long max, long seed);
void        row_start(int t);
void        row_stop(int t);
void        dump_seeds(int t);

/* text.c */
#define MAX_GRAMMAR_LEN             12          /* max length of
grammar component */
#define MAX_SENT_LEN        256 /* max length of populated
sentence */
#define RNG_PER_SENT        27          /* max number of RNG
calls per sentence */

int                 dbg_text PROTO((char * t, int min, int max, int
s));

#ifdef DECLARER
#define EXTERN
#else
#define EXTERN extern
#endif          /* DECLARER */


EXTERN distribution nations;
EXTERN distribution nations2;
EXTERN distribution regions;
EXTERN distribution o_priority_set;
EXTERN distribution l_instruct_set;
EXTERN distribution l_smode_set;
EXTERN distribution l_category_set;
EXTERN distribution l_rflag_set;
EXTERN distribution c_mseg_set;
EXTERN distribution colors;
EXTERN distribution p_types_set;
EXTERN distribution p_cntr_set;

/* distributions that control text generation */
EXTERN distribution articles;
EXTERN distribution nouns;
EXTERN distribution adjectives;
EXTERN distribution adverbs;
EXTERN distribution prepositions;
EXTERN distribution verbs;
EXTERN distribution terminators;
EXTERN distribution auxillaries;
EXTERN distribution np;
EXTERN distribution vp;
EXTERN distribution grammar;


EXTERN long scale;
EXTERN int refresh;
EXTERN int resume;
EXTERN long verbose;
EXTERN long force;
EXTERN long header;
EXTERN long columnar;
EXTERN long direct;
EXTERN long updates;
EXTERN long table;
EXTERN long children;
EXTERN long fnames;
EXTERN int  gen_sql;
EXTERN int  gen_rng;
EXTERN char *db_name;
EXTERN int  step;
EXTERN int              set_seeds;
EXTERN int  validate;
EXTERN char *d_path;

/* added for segmented updates */
EXTERN int insert_segments;
EXTERN int delete_segments;
EXTERN int insert_orders_segment;
EXTERN int insert_lineitem_segment;
EXTERN int delete_segment;


#ifndef DECLARER
extern tdef tdefs[];

#endif          /* DECLARER */


/*************************************************************
*******
 ** table level defines use the following naming convention: t_ccc_xxx
 ** with: t, a table identifier
 **       ccc, a column identifier
 **       xxx, a limit type

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

/*
 * defines which control the parts table
 */
#define  P_SIZE      126
#define  P_NAME_SCL   5
#define  P_MFG_TAG    "Manufacturer#"
#define  P_MFG_FMT    "%s%01d"
#define  P_MFG_MIN    1
#define  P_MFG_MAX    5
#define  P_BRND_TAG   "Brand#"
#define  P_BRND_FMT   "%s%02d"
#define  P_BRND_MIN    1
#define  P_BRND_MAX    5
#define  P_SIZE_MIN   1
#define  P_SIZE_MAX   50
#define  P_MCST_MIN   100
#define  P_MCST_MAX   99900
#define  P_MCST_SCL   100.0
#define  P_RCST_MIN   90000
#define  P_RCST_MAX   200000
#define  P_RCST_SCL   100.0
/*
 * defines which control the suppliers table
```

```
 */
#define  S_SIZE     145
#define  S_NAME_TAG "Supplier#"
#define  S_NAME_FMT "%s%09ld"
#define  S_ABAL_MIN   -99999
#define  S_ABAL_MAX    999999
#define  S_CMNT_MAX    101
#define  S_CMNT_BBB    10       /* number of BBB comments/SF */
#define  BBB_DEADBEATS 50       /* % that are complaints */
#define  BBB_BASE  "Customer "
#define  BBB_COMPLAIN  "Complaints"
#define  BBB_COMMEND   "Recommends"
#define  BBB_CMNT_LEN  19
#define  BBB_BASE_LEN  9
#define  BBB_TYPE_LEN  10

/*
 * defines which control the partsupp table
 */
#define  PS_SIZE      145
#define  PS_SKEY_MIN  0
#define  PS_SKEY_MAX  ((tdefs[SUPP].base - 1) * scale)
#define  PS_SCST_MIN  100
#define  PS_SCST_MAX  100000
#define  PS_QTY_MIN   1
#define  PS_QTY_MAX   9999
/*
 * defines which control the customers table
 */
#define  C_SIZE       165
#define  C_NAME_TAG   "Customer#"
#define  C_NAME_FMT   "%s%09d"
#define  C_MSEG_MAX    5
#define  C_ABAL_MIN   -99999
#define  C_ABAL_MAX    999999
/*
 * defines which control the order table
 */
#define  O_SIZE         109
#define  O_CKEY_MIN      1
#define  O_CKEY_MAX      (long)(tdefs[CUST].base * scale)
#define  O_ODATE_MIN     STARTDATE
#define  O_ODATE_MAX     (STARTDATE + TOTDATE - \
                (L_SDTE_MAX + L_RDTE_MAX) - 1)
#define  O_CLRK_TAG      "Clerk#"
#define  O_CLRK_FMT      "%s%09d"
#define  O_CLRK_SCL      1000
#define  O_LCNT_MIN      1
#define  O_LCNT_MAX      7

/*
 * defines which control the lineitem table
 */
#define  L_SIZE       144L
#define  L_QTY_MIN    1
#define  L_QTY_MAX    50
#define  L_TAX_MIN    0
#define  L_TAX_MAX    8
#define  L_DCNT_MIN   0
#define  L_DCNT_MAX   10
#define  L_PKEY_MIN   1
#define  L_PKEY_MAX   (tdefs[PART].base * scale)
#define  L_SDTE_MIN   1
#define  L_SDTE_MAX   121
#define  L_CDTE_MIN   30
#define  L_CDTE_MAX   90
#define  L_RDTE_MIN   1
#define  L_RDTE_MAX   30
/*
 * defines which control the time table
 */
#define  T_SIZE       30
#define  T_START_DAY  3     /* wednesday ? */
#define  LEAP(y)  ((!(y % 4) && (y % 100))?1:0)

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

*************************************************************
********
 ***
 *** general or inter table defines
 ***

*************************************************************
********

*************************************************************
********/
#define  SUPP_PER_PART 4
#define  ORDERS_PER_CUST 10 /* sync this with
CUST_MORTALITY */
#define  CUST_MORTALITY 3  /* portion with have no orders */
#define  NATIONS_MAX  90 /* limited by country codes in phone
numbers */
#define  PHONE_FMT    "%02d-%03d-%03d-%04d"
#define  STARTDATE    92001
#define  CURRENTDATE  95168
#define  ENDDATE      98365
#define  TOTDATE      2557
#define  UPD_PCT      10
#define  MAX_STREAM   47
#define  V_STR_LOW    0.4
#define  PENNIES    100 /* for scaled int money arithmetic */
#define  Q11_FRACTION (double)0.0001
/*
 * max and min SF in GB; Larger SF will require changes to the build
routines
 */
#define  MIN_SCALE     1.0
#define  MAX_SCALE   3000.0
/*
 * beyond this point we need to allow for BCD calculations
 */
#define  MAX_32B_SCALE   3000.0
#define INIT_HUGE(v)            { \
                                v = (DSS_HUGE
*)malloc(sizeof(DSS_HUGE) * HUGE_COUNT); \
                                MALLOC_CHECK(v); \
                                }
#define FREE_HUGE(v)            free(v)
#ifdef SUPPORT_64BITS
#define LONG2HUGE(src, dst)             *dst = (DSS_HUGE)src

#define HUGE2LONG(src, dst)             *dst = (long)src
#define HUGE_SET(src, dst)              *dst = *src
#define HUGE_MUL(op1, op2)              *op1 *= op2
#define HUGE_DIV(op1, op2)              *op1 /= op2
#define HUGE_ADD(op1, op2, dst)         *dst = *op1 + op2
#define HUGE_SUB(op1, op2, dst)         *dst = *op1 - op2
#define HUGE_MOD(op1, op2)              *op1 % op2
#define HUGE_CMP(op1, op2)              (*op1 == *op2)?0:(*op1 <
*op2)-1:1
#else
#define LONG2HUGE(src, dst)             {*dst = src; *(dst + 1) = 0;}
#define HUGE2LONG(src, dst)             { dst=0 ; \
                                                bcd2_bin(dst,
(src + 1)); \
                                                bcd2_bin(dst,
src); }
#define HUGE_SET(src, dst)              { *dst = *src ; *(dst + 1) =
*(src + 1); }
```

```
#define HUGE_MUL(op1,op2)               bcd2_mul(op1, (op1 + 1),
op2)
#define HUGE_DIV(op1,op2)               bcd2_div(op1, (op1 + 1),
op2)
#define HUGE_ADD(op1,op2,d)             { \

        HUGE_SET(op1, d); \
                                                bcd2_add(d, (d
+ 1), op2); \
                                                }
#define HUGE_SUB(op1,op2,d)             { \

        HUGE_SET(op1, d); \
                                                bcd2_sub(d, (d
+ 1), op2); \
                                                }
#define HUGE_MOD(op1, op2)              bcd2_mod(op1, (op1 + 1),
op2)
#define HUGE_CMP(op1, op2)              (bcd2_cmp(op1, (op1 + 1),
op2) == 0)?0:\

((bcd2_cmp(op1, (op1 + 1), op2) < 0)?-1:1)
#endif /* SUPPORT_64BITS */

/******** environmental variables and defaults ***************/
#define  DIST_TAG  "DSS_DIST"                   /* environment
var to override ... */
#define  DIST_DFLT "dists.dss"          /* default file to hold
distributions */
#define  PATH_TAG  "DSS_PATH"                   /* environment
var to override ... */
#define  PATH_DFLT "."                                  /*
default directory to hold tables */
#define  CONFIG_TAG  "DSS_CONFIG"   /* environment var to
override ... */
#define  CONFIG_DFLT "."                        /* default
directory to config files */
#define  ADHOC_TAG  "DSS_ADHOC"                 /* environment
var to override ... */
#define  ADHOC_DFLT "adhoc.dss"                 /* default file
name for adhoc vars */

/******* output macros ********/
#ifndef SEPARATOR
#define SEPARATOR '|' /* field spearator for generated flat files */
#endif
/* Data type flags for a single print routine */
#define DT_STR                  0
#ifndef MVS
#define DT_VSTR                 DT_STR
#else
#define DT_VSTR                 1
#endif /* MVS */
#define DT_INT                  2
#define DT_HUGE                 3
#define DT_KEY                  4
#define DT_MONEY 5
#define DT_CHR                  6

int dbg_print(int dt, FILE *tgt, void *data, int len, int eol);
#define PR_STR(fp, str, len)            dbg_print(DT_STR, fp,
(void *)str, len, 1)
#define PR_VSTR(fp, str, len)   dbg_print(DT_VSTR, fp, (void *)str,
len, 1)
#define PR_VSTR_LAST(fp, str, len)      dbg_print(DT_VSTR, fp,
(void *)str, len, 0)
#define PR_INT(fp, str)
            dbg_print(DT_INT, fp, (void *)str, 0, 1)
#define PR_HUGE(fp, str)                dbg_print(DT_HUGE, fp,
(void *)str, 0, 1)
#define PR_KEY(fp, str)
            dbg_print(DT_KEY, fp, (void *)str, 0, -1)
#define PR_MONEY(fp, str)               dbg_print(DT_MONEY,
fp, (void *)str, 0, 1)
#ifdef LINUX
  #define PR_CHR(fp, str)
            dbg_print(DT_CHR, fp, str, 0, 1)
#else
  #define PR_CHR(fp, str)
            dbg_print(DT_CHR, fp, (void *)str, 0, 1)
#endif
#define  PR_STRT(fp)   /* any line prep for a record goes here */
#define  PR_END(fp)    fprintf(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 */

/*
 * verification macros
 */
#define  VRF_STR(t, d) {char *xx = d; while (*xx) tdefs[t].vtotal +=
*xx++;}
#define  VRF_INT(t,d)  tdefs[t].vtotal += d
#ifdef SUPPORT_64BITS
#define  VRF_HUGE(t,d)          tdefs[t].vtotal = *((long *)&d) +
*((long *)(&d + 1))
#else
#define VRF_HUGE(t,d)           tdefs[t].vtotal += d[0] + d[1]
#endif /* SUPPORT_64BITS */
/* assume float is a 64 bit quantity */
#define  VRF_MONEY(t,d)         tdefs[t].vtotal = *((long *)&d) +
*((long *)(&d + 1))
#define  VRF_CHR(t,d)           tdefs[t].vtotal += d
#define  VRF_STRT(t)
#define  VRF_END(t)

/*********** distribuitons currently defined *************/
#define  UNIFORM   0

/*
 * seed indexes; used to separate the generation of individual columns
 */
#define  P_MFG_SD  0
#define  P_BRND_SD 1
#define  P_TYPE_SD 2
#define  P_SIZE_SD 3
#define  P_CNTR_SD 4
#define  P_RCST_SD 5
#define  PS_QTY_SD 7
#define  PS_SCST_SD   8
#define  O_SUPP_SD 10
#define  O_CLRK_SD 11
#define  O_ODATE_SD   13
#define  L_QTY_SD  14
#define  L_DCNT_SD 15
#define  L_TAX_SD  16
#define  L_SHIP_SD 17
#define  L_SMODE_SD   18
#define  L_PKEY_SD 19
#define  L_SKEY_SD 20
#define  L_SDTE_SD 21
#define  L_CDTE_SD 22
#define  L_RDTE_SD 23
#define  L_RFLG_SD 24
#define  C_NTRG_SD 27
#define  C_PHNE_SD 28
#define  C_ABAL_SD 29
#define  C_MSEG_SD 30
#define  S_NTRG_SD 33
#define  S_PHNE_SD 34
#define  S_ABAL_SD 35
```

```
#define  P_NAME_SD 37
#define  O_PRIO_SD 38
#define  HVAR_SD   39
#define  O_CKEY_SD 40
#define  N_CMNT_SD 41
#define  R_CMNT_SD 42
#define  O_LCNT_SD 43
#define  BBB_JNK_SD    44
#define  BBB_TYPE_SD   45
#define  BBB_CMNT_SD   46
#define  BBB_OFFSET_SD 47
#endif           /* DSS_H */
```

dsstypes.h

```
/*
 * $Header: dsstypes.h 11-apr-2001.14:58:22 mpoess Exp $
 */

/* Copyright (c) Oracle Corporation 1999, 2000. All Rights Reserved. */

/* NOTE:  See 'header_template.doc' in the 'doc' dve under the 'forms'
      directory for the header file template that includes instructions.
*/

/*
   NAME
     <<<LOWERCASE_FILENAME>>>.h - <one-line expansion of the
name>

   DESCRIPTION
     <short description of component this file declares/defines>

   RELATED DOCUMENTS

   INSPECTION STATUS
     Inspection date:
     Inspection status:
     Estimated increasing cost defects per page:
     Rule sets:

   ACCEPTANCE REVIEW STATUS
     Review date:
     Review status:
     Reviewers:

   PUBLIC FUNCTION(S)
     <list of external functions declared/defined - with one-line
descriptions>

   PRIVATE FUNCTION(S)
     <list of static functions defined in .c file - with one-line
descriptions>

   EXAMPLES

   NOTES
     <other useful comments, qualifications, etc.>

   MODIFIED   (MM/DD/YY)
   mpoess      04/11/01 -
   mpoess      08/28/00 - add support for SF>1000
   mpoess      07/12/99 - exchange with new version July 8 1999
   mpoess      06/28/99 - Creation
   mpoess      06/28/99 - Creation

*/
 /*
 * Sccsid:     @(#)dsstypes.h        2.1.8.1
 *
 * general definitions and control information for the DSS data types
 * and function prototypes
 */

/*
 * typedefs
 */
typedef struct
{
    long          custkey;
    char          name[C_NAME_LEN + 1];
    char          address[C_ADDR_MAX + 1];
    int           alen;
    long          nation_code;
    char          phone[PHONE_LEN + 1];
    long          acctbal;
    char          mktsegment[MAXAGG_LEN + 1];
    char          comment[C_CMNT_MAX + 1];
    int           clen;
}             customer_t;
/* customers.c */
long mk_cust   PROTO((long n_cust, customer_t * c));
int pr_cust    PROTO((customer_t * c, int mode));
int ld_cust    PROTO((customer_t * c, int mode));

typedef struct
{
    DSS_HUGE              *okey;
    long          partkey;
    long          suppkey;
    long          lcnt;
    long          quantity;
    long          eprice;
    long          discount;
    long          tax;
    char          rflag[1];
    char          lstatus[1];
    char          cdate[DATE_LEN];
    char          sdate[DATE_LEN];
    char          rdate[DATE_LEN];
    char          shipinstruct[MAXAGG_LEN + 1];
    char          shipmode[MAXAGG_LEN + 1];
    char          comment[L_CMNT_MAX + 1];
    int           clen;
}             line_t;

typedef struct
{
    DSS_HUGE              *okey;
    long          custkey;
    char          orderstatus;
    long          totalprice;
    char          odate[DATE_LEN];
    char          opriority[MAXAGG_LEN + 1];
    char          clerk[O_CLRK_LEN + 1];
    long          spriority;
    long          lines;
    char          comment[O_CMNT_MAX + 1];
    int           clen;
    line_t        l[O_LCNT_MAX];
}             order_t;

/* order.c */
long          mk_order  PROTO((DSS_HUGE index, order_t * o, long
upd_num));
int                     pr_order    PROTO((order_t * o, int mode));
int                     ld_order    PROTO((order_t * o, int mode));
void          ez_sparse  PROTO((DSS_HUGE index, DSS_HUGE *ok,
long seq));
#ifndef SUPPORT_64BITS
void          hd_sparse  PROTO((long index, DSS_HUGE *ok, long
seq));
#endif
```

```
typedef struct
{
   long          partkey;
   long          suppkey;
   long          qty;
   long          scost;
   char          comment[PS_CMNT_MAX + 1];
   int           clen;
}             partsupp_t;

typedef struct
{
   long          partkey;
   char          name[P_NAME_LEN + 1];
   int           nlen;
   char          mfgr[P_MFG_LEN + 1];
   char          brand[P_BRND_LEN + 1];
   char          type[P_TYPE_LEN + 1];
   int           tlen;
   long          size;
   char          container[P_CNTR_LEN + 1];
   long          retailprice;
   char          comment[P_CMNT_MAX + 1];
   int           clen;
   partsupp_t    s[SUPP_PER_PART];
}             part_t;

/* parts.c */
long mk_part   PROTO((long index, part_t * p));
int pr_part    PROTO((part_t * part, int mode));
int ld_part    PROTO((part_t * part, int mode));

typedef struct
{
   long          suppkey;
   char          name[S_NAME_LEN + 1];
   char          address[S_ADDR_MAX + 1];
   int           alen;
   long          nation_code;
   char          phone[PHONE_LEN + 1];
   long          acctbal;
   char          comment[S_CMNT_MAX + 1];
   int           clen;
}             supplier_t;
/* supplier.c */
long mk_supp   PROTO((long index, supplier_t * s));
int pr_supp    PROTO((supplier_t * supp, int mode));
int ld_supp    PROTO((supplier_t * supp, int mode));

typedef struct
{
   long          timekey;
   char          alpha[DATE_LEN];
   long          year;
   long          month;
   long          week;
   long          day;
} dss_time_t;

/* time.c */
long mk_time   PROTO((long index, dss_time_t * t));

/*
 * this assumes that N_CMNT_LEN >= R_CMNT_LEN
 */
typedef struct
{
   long          code;
   char          *text;
   long          join;
   char          comment[N_CMNT_MAX + 1];
   int           clen;
}             code_t;

/* code table */
int mk_nation   PROTO((long i, code_t * c));
int pr_nation   PROTO((code_t * c, int mode));
int ld_nation   PROTO((code_t * c, int mode));
int mk_region   PROTO((long i, code_t * c));
int pr_region   PROTO((code_t * c, int mode));
int ld_region   PROTO((code_t * c, int mode));
```

load_stub.c

```
#ifdef RCSID
static char *RCSid =
  "$Header: load_stub.c 11-apr-2001.14:56:58 mpoess Exp $ ";
#endif /* RCSID */

/* Copyright (c) Oracle Corporation 1999. All Rights Reserved. */

/*

  NAME
    load_stub.c - <one-line expansion of the name>

  DESCRIPTION
    <short description of component this file declares/defines>

  PUBLIC FUNCTION(S)
    <list of external functions declared/defined - with one-line
descriptions>

  PRIVATE FUNCTION(S)
    <list of static functions defined in .c file - with one-line
descriptions>

  RETURNS
    <function return values, for .c file with single function>

  NOTES
    <other useful comments, qualifications, etc.>

  MODIFIED   (MM/DD/YY)
  mpoess      04/11/01 -
  mpoess      07/12/99 - exchange with new version July 8 1999
  mpoess      06/28/99 - Creation
  mpoess      06/28/99 - Creation

*/

/**************************************************************
*******
 *  Title:      load_stub.c
 *  Sccsid:     @(#)load_stub.c     2.1.8.1
 *  Description:
 *              stub routines for:
 *          inline load of dss benchmark
 *          header creation for dss benchmark
 *

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

#include <stdio.h>
#include "config.h"
#include "dss.h"
#include "dsstypes.h"

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

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

   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)
{
   static int count = 0;

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

   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_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 *p, int mode)
{
   static int count = 0;

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

   return(0);
}

ld_line (order_t *p, int mode)
{
   static int count = 0;

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

   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

```
#ifdef RCSID
static char *RCSid =
  "$Header: part.c 01-dec-99.16:46:46 mpoess Exp $ ";
#endif /* RCSID */

/* Copyright (c) Oracle Corporation 1998, 1999. All Rights Reserved. */

/*

  NAME
    part.c - <one-line expansion of the name>

  DESCRIPTION
    <short description of component this file declares/defines>

  PUBLIC FUNCTION(S)
    <list of external functions declared/defined - with one-line
descriptions>

  PRIVATE FUNCTION(S)
    <list of static functions defined in .c file - with one-line
descriptions>

  RETURNS
    <function return values, for .c file with single function>

  NOTES
    <other useful comments, qualifications, etc.>

  MODIFIED   (MM/DD/YY)
  mpoess      12/01/99 - Oracle functions for partitioning

*/

/*=================================================
=================+
|      Copyright (c) 1997 , 1999 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 "dss.h"
#include "dsstypes.h"
#include <string.h>
#include "part.h"

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

#ifdef DEBUG
extern long row_count;
extern long *part_row;
extern int step;
extern FILE* debug_fd;
#endif /* DEBUG */

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 = (FILE **) malloc(numpart * sizeof(FILE *));

#ifdef DEBUG
  part_row = (long *) malloc(sizeof(long) * numpart);
#endif /* DEBUG */
  /* 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) == EOF) {
                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] = fopen(ofile, "w");
    if (out[j] == NULL) {
      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 partitons.\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(errn)
     int errn;
{
  int i,j;

  /* 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++)
    fclose(out[i]);
```

```
#ifdef DEBUG
  if (debug_fd != NULL) {
    fprintf(debug_fd, "dbgen was unsuccessful for step %d\n", step);
    fprintf(debug_fd, "total number of rows generated: %ld\n", (long)
row_count);
    for(j=0;j<numpart;j++) {
      fprintf(debug_fd, "partition: %d rows: %ld\n",
                    j+1, part_row[j]);
    }
    free(part_row);
    fclose(debug_fd);
  }
#endif /* DEBUG */

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


int find_partition(cval,type)
     void *cval;
     int type;
{
  char tmpdate[16];
  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:
    strcpy(tmpdate, (char*)cval);
    ASS_DATE(vdat,(char *)tmpdate);
    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++;
  }
}

int find_partition_cust(customer_t *c)
{
double tmpnum;
   switch (values->pos)
   {
      case 1:
         tmpnum = c->custkey;
         return(find_partition(&(tmpnum), NUM_TYP));
         break;
      case 2:
         return(find_partition(c->mktsegment, STR_TYP));
         break;
      case 3:
         tmpnum = c->nation_code;
         return(find_partition(&(tmpnum), NUM_TYP));
         break;
      case 4:
         return(find_partition(c->name, STR_TYP));
         break;
      case 5:
         return(find_partition(c->address, STR_TYP));
         break;
      case 6:
         return(find_partition(c->phone, STR_TYP));
         break;
      case 7:
         tmpnum = c->acctbal;
         return(find_partition(&(tmpnum), NUM_TYP));
         break;
      case 8:
         return(find_partition(c->comment, STR_TYP));
         break;
      default:
         fprintf(stderr,"Incorrect column number for partitioning.\n");
         Exitprog(-1);
         break;
   }
}

int find_partition_order(order_t *o)
{
double tmpnum;
   switch (values->pos)
   {
      case 1:
         return(find_partition(o->odate, DAT_TYP));
         break;
      case 2:
         tmpnum = *(o->okey);
         return(find_partition(&(tmpnum), NUM_TYP));
         break;
      case 3:
         tmpnum = o->custkey;
         return(find_partition(&(tmpnum), NUM_TYP));
         break;
      case 4:
         return(find_partition(o->opriority, STR_TYP));
         break;
      case 5:
         tmpnum = o->spriority;
         return(find_partition(&(tmpnum), NUM_TYP));
         break;
      case 6:
         return(find_partition(o->clerk, STR_TYP));
         break;
      case 7:
         return(find_partition(o->orderstatus, STR_TYP));
         break;
      case 8:
         tmpnum = o->totalprice;
         return(find_partition(&(tmpnum), NUM_TYP));
         break;
      case 9:
         return(find_partition(o->comment, STR_TYP));
         break;
      default:
         fprintf(stderr,"Incorrect column number for partitioning.\n");
         Exitprog(-1);
         break;
   }
}

int find_partition_line(order_t *o, int i)
{
double tmpnum;

   switch (values->pos)
   {
      case 1:
         return(find_partition(o->l[i].sdate, DAT_TYP));
         break;
      case 2:
         tmpnum = *(o->okey);
         return(find_partition(&(tmpnum), NUM_TYP));
         break;
      case 3:
         tmpnum = o->l[i].discount;
         return(find_partition(&(tmpnum), NUM_TYP));
         break;
      case 4:
         tmpnum = o->l[i].eprice;
         return(find_partition(&(tmpnum), NUM_TYP));
         break;
      case 5:
         tmpnum = o->l[i].suppkey;
         return(find_partition(&(tmpnum), NUM_TYP));
```

```
            break;
        case 6:
            tmpnum = o->l[i].quantity;
            return(find_partition(&(tmpnum), NUM_TYP));
            break;
        case 7:
            return(find_partition(o->l[i].rflag[0], STR_TYP));
            break;
        case 8:
            tmpnum = o->l[i].partkey;
            return(find_partition(&(tmpnum), NUM_TYP));
            break;
        case 9:
            return(find_partition(o->l[i].lstatus[0], STR_TYP));
            break;
        case 10:
            tmpnum = o->l[i].tax;
            return(find_partition(&(tmpnum), NUM_TYP));
            break;
        case 11:
            return(find_partition(o->l[i].cdate, DAT_TYP));
            break;
        case 12:
            return(find_partition(o->l[i].rdate, DAT_TYP));
            break;
        case 13:
            return(find_partition(o->l[i].shipmode, STR_TYP));
            break;
        case 14:
            tmpnum = o->l[i].lcnt;
            return(find_partition(&(tmpnum), NUM_TYP));
            break;
        case 15:
            return(find_partition(o->l[i].shipinstruct, STR_TYP));
            break;
        case 16:
            return(find_partition(o->l[i].comment, STR_TYP));
            break;
        default:
            fprintf(stderr,"Incorrect column number for partitioning.\n");
            Exitprog(-1);
            break;
    }
}

int find_partition_part(part_t *p)
{
double tmpnum;

    switch (values->pos)
    {
        case 1:
            tmpnum = p->partkey;
            return(find_partition(&(tmpnum), NUM_TYP));
            break;
        case 2:
            return(find_partition(p->type, STR_TYP));
            break;
        case 3:
            tmpnum = p->size;
            return(find_partition(&(tmpnum), NUM_TYP));
            break;
        case 4:
            return(find_partition(p->brand, STR_TYP));
            break;
        case 5:
            return(find_partition(p->name, STR_TYP));
            break;
        case 6:
            return(find_partition(p->container, STR_TYP));
            break;
        case 7:
            return(find_partition(p->mfgr, STR_TYP));
            break;
        case 8:
            tmpnum = p->retailprice;
            return(find_partition(&(tmpnum), NUM_TYP));
            break;
        case 9:
            return(find_partition(p->comment, STR_TYP));
            break;
        default:
            fprintf(stderr,"Incorrect column number for partitioning.\n");
            Exitprog(-1);
            break;
    }
}

int find_partition_psupp(partsupp_t *ps)
{
double tmpnum;

    switch (values->pos)
    {
        case 1:
            tmpnum = ps->partkey;
            return(find_partition(&(tmpnum), NUM_TYP));
            break;
        case 2:
            tmpnum = ps->suppkey;
            return(find_partition(&(tmpnum), NUM_TYP));
            break;
        case 3:
            tmpnum = ps->qty;
            return(find_partition(&(tmpnum), NUM_TYP));
            break;
        case 4:
            tmpnum = ps->scost;
            return(find_partition(&(tmpnum), NUM_TYP));
            break;
        case 5:
            return(find_partition(ps->comment, STR_TYP));
            break;
        default:
            fprintf(stderr,"Incorrect column number for partitioning.\n");
            Exitprog(-1);
            break;
    }
}

int find_partition_supp(supplier_t *s)
{
double tmpnum;

    switch (values->pos)
    {
        case 1:
            tmpnum = s->suppkey;
            return(find_partition(&(tmpnum), NUM_TYP));
            break;
        case 2:
            tmpnum = s->nation_code;
            return(find_partition(&(tmpnum), NUM_TYP));
            break;
        case 3:
            return(find_partition(s->comment, STR_TYP));
            break;
        case 4:
            return(find_partition(s->name, STR_TYP));
            break;
        case 5:
            return(find_partition(s->address, STR_TYP));
            break;
        case 6:
            return(find_partition(s->phone, STR_TYP));
            break;
```

```
        case 7:
            tmpnum = s->acctbal;
            return(find_partition(&(tmpnum), NUM_TYP));
            break;
        default:
            fprintf(stderr,"Incorrect column number for partitioning.\n");
            Exitprog(-1);
            break;
    }
}
```

part.h

```
/*
 * $Header: part.h 01-dec-99.16:48:14 mpoess Exp $
 */

/* Copyright (c) Oracle Corporation 1998, 1999. All Rights Reserved. */

/* NOTE:  See 'header_template.doc' in the 'doc' dve under the 'forms'
      directory for the header file template that includes instructions.
*/

/*
   NAME
     part.h - <one-line expansion of the name>

   DESCRIPTION
     <short description of component this file declares/defines>

   RELATED DOCUMENTS

   INSPECTION STATUS
     Inspection date:
     Inspection status:
     Estimated increasing cost defects per page:
     Rule sets:

   ACCEPTANCE REVIEW STATUS
     Review date:
     Review status:
     Reviewers:

   PUBLIC FUNCTION(S)
     <list of external functions declared/defined - with one-line
descriptions>

   PRIVATE FUNCTION(S)
     <list of static functions defined in .c file - with one-line
descriptions>

   EXAMPLES

   NOTES
     <other useful comments, qualifications, etc.>

   MODIFIED   (MM/DD/YY)
   mpoess      12/01/99 - Oracle changes for partitioning

*/


#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <errno.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 spearator 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(fp,flt)    \
        if (columnar) sprintf(fp, MONEY_COL_FMT, \
              flt/100, ((flt < 0)?-flt:flt)%100); \
        else sprintf(fp,"%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();
```

permute.c

```
#ifdef RCSID
static char *RCSid =
  "$Header: permute.c 11-apr-2001.14:56:12 mpoess Exp $ ";
#endif /* RCSID */

/* Copyright (c) Oracle Corporation 1999. All Rights Reserved. */

/*

  NAME
    permute.c - <one-line expansion of the name>

  DESCRIPTION
    <short description of component this file declares/defines>

  PUBLIC FUNCTION(S)
    <list of external functions declared/defined - with one-line
descriptions>

  PRIVATE FUNCTION(S)
    <list of static functions defined in .c file - with one-line
descriptions>

  RETURNS
    <function return values, for .c file with single function>

  NOTES
    <other useful comments, qualifications, etc.>

  MODIFIED   (MM/DD/YY)
  mpoess     04/11/01 -
  mpoess     07/12/99 - exchange with new version July 8 1999
  mpoess     06/28/99 - Creation
  mpoess     06/28/99 - Creation

*/

/* @(#)permute.c       2.1.8.3 */
/*
* permute.c -- a permutation generator for the query
*             sequences in TPC-H and TPC-R
*/

#ifdef TEST
#define DECLARER
#endif
#include "config.h"
#include "dss.h"
#ifdef TEST
#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>
#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 NOSYSMETRICS
#define NOTEXTMETRIC
#define NOWH
#define NOCOMM
#define NOKANJI
#define NOMCX
#include <windows.h>
#pragma warning(default:4201)
#pragma warning(default:4214)
#endif
#endif

long NextRand(long seed);
long *permute(long *set, int cnt, long stream);
long *permute_dist(distribution *d, long stream);
long seed;
char *eol[2] = {" ", "},"};
extern seed_t Seed[];
#ifdef TEST
tdef tdefs = { NULL };
#endif


#define MAX_QUERY               22
#define ITERATIONS              1000
#define UNSET           0

long *
permute(long *a, int c, long s)
                {
    int i;
    static long source;
    static long *set, temp;

                if (a != (long *)NULL)
                                {
                                set = a;
                                for (i=0; i < c; i++)
                                                *(a + i) = i;
                                for (i=0; i < c; i++)
                                                {
                                                RANDOM(source, 0L, (long)(c - 1),
s);
                                                temp = *(a + source);
                                                *(a + source) = *(a + i) ;
                                                *(a + i) = temp;
```

```
                                source = 0;
                                }
                        }
        else
                        source += 1;

        if (source >= c)
                        source -= c;

        return(set + source);
        }

long *
permute_dist(distribution *d, long stream)
        {
        static distribution *dist = NULL;
        int i;

        if (d != NULL)
                        {
                        if (d->permute == (long *)NULL)
                                        {
                                        d->permute = (long
*)malloc(sizeof(long) * DIST_SIZE(d));
                                        MALLOC_CHECK(d->permute);
                                        for (i=0; i < DIST_SIZE(d); i++)
                                                        *(d->permute + i) = i;
                                        }
                        dist = d;
                        return(permute(dist->permute, DIST_SIZE(dist),
stream));
                        }


        if (dist != NULL)
                        return(permute(NULL, DIST_SIZE(dist),
stream));
        else
                        INTERNAL_ERROR("Bad call to
permute_dist");
        }


#ifdef TEST

main(int ac, char *av[])
        {
        long *sequence,
                        i,
                        j,
                        streams = UNSET,
                        *a;
        char sep;
        int index = 0;

        set_seeds = 0;
        sequence = (long *)malloc(MAX_QUERY * sizeof(long));
        a = sequence;
        for (i=0; i < MAX_QUERY; i++)
                        *(sequence + i) = i;
        if (ac < 3)
                        goto usage;
        Seed[0].value = (long)atoi(av[1]);
        streams = atoi(av[2]);
        if (Seed[0].value == UNSET || streams == UNSET)
                        goto usage;

        index = 0;
        printf("long permutation[%d][%d] = {\n", streams,
MAX_QUERY);
        for (j=0; j < streams; j++)
                        {
                        sep = '{';
                        printf("%s\n", eol[index]);
                        for (i=0; i < MAX_QUERY; i++)
                                        {
                                        printf("%c%2d", sep, *permute(a,
MAX_QUERY, 0) + 1);
                                        a = (long *)NULL;
                                        sep = ',';
                                        }
                        a = sequence;
                        index=1;
                        }
        printf("}\n};\n");
        return(0);

usage:
        printf("Usage: %s <seed> <streams>\n",av[0]);
        printf("  uses <seed> to start the generation of <streams>
permutations of [1..%d]\n", MAX_QUERY);
        return(-1);

        }
#endif /* TEST */
```

permute.h

```
/*
 * $Header: permute.h 11-apr-2001.14:55:31 mpoess Exp $
 */

/* Copyright (c) Oracle Corporation 1999. All Rights Reserved. */

/* NOTE:  See 'header_template.doc' in the 'doc' dve under the 'forms'
     directory for the header file template that includes instructions.
*/

/*
  NAME
    <<<LOWERCASE_FILENAME>>>.h - <one-line expansion of the
name>

  DESCRIPTION
    <short description of component this file declares/defines>

  RELATED DOCUMENTS

  INSPECTION STATUS
    Inspection date:
    Inspection status:
    Estimated increasing cost defects per page:
    Rule sets:

  ACCEPTANCE REVIEW STATUS
    Review date:
    Review status:
    Reviewers:

  PUBLIC FUNCTION(S)
    <list of external functions declared/defined - with one-line
descriptions>

  PRIVATE FUNCTION(S)
    <list of static functions defined in .c file - with one-line
descriptions>

  EXAMPLES

  NOTES
    <other useful comments, qualifications, etc.>

  MODIFIED   (MM/DD/YY)
```

```
   mpoess     04/11/01 -
   mpoess     07/12/99 - exchange with new version July 8 1999
   mpoess     06/28/99 - Creation
   mpoess     06/28/99 - Creation

*/
/*
 * @(#)permute.h       2.1.8.1
 */
long permutation[41][22] =
{
  {14, 2, 9,20, 6,17,18, 8,21,13, 3,22,16, 4,11,15, 1,10,19, 5, 7,12},
  {21, 3,18, 5,11, 7, 6,20,17,12,16,15,13,10, 2, 8,14,19, 9,22, 1, 4},
  { 6,17,14,16,19,10, 9, 2,15, 8, 5,22,12, 7,13,18, 1, 4,20, 3,11,21},
  { 8, 5, 4, 6,17, 7, 1,18,22,14, 9,10,15,11,20, 2,21,19,13,16,12, 3},
  { 5,21,14,19,15,17,12, 6, 4, 9, 8,16,11, 2,10,18, 1,13, 7,22, 3,20},
  {21,15, 4, 6, 7,16,19,18,14,22,11,13, 3, 1, 2, 5, 8,20,12,17,10, 9},
  {10, 3,15,13, 6, 8, 9, 7, 4,11,22,18,12, 1, 5,16, 2,14,19,20,17,21},
  {18, 8,20,21, 2, 4,22,17, 1,11, 9,19, 3,13, 5, 7,10,16, 6,14,15,12},
  {19, 1,15,17, 5, 8, 9,12,14, 7, 4, 3,20,16, 6,22,10,13, 2,21,18,11},
  { 8,13, 2,20,17, 3, 6,21,18,11,19,10,15, 4,22, 1, 7,12, 9,14, 5,16},
  { 6,15,18,17,12, 1, 7, 2,22,13,21,10,14, 9, 3,16,20,19,11, 4, 8, 5},
  {15,14,18,17,10,20,16,11, 1, 8, 4,22, 5,12, 3, 9,21, 2,13, 6,19, 7},
  { 1, 7,16,17,18,22,12, 6, 8, 9,11, 4, 2, 5,20,21,13,10,19, 3,14,15},
  {21,17, 7, 3, 1,10,12,22, 9,16, 6,11, 2, 4, 5,14, 8,20,13,18,15,19},
  { 2, 9, 5, 4,18, 1,20,15,16,17, 7,21,13,14,19, 8,22,11,10, 3,12, 6},
  {16, 9,17, 8,14,11,10,12, 6,21, 7, 3,15, 5,22,20, 1,13,19, 2, 4,18},
  { 1, 3, 6, 5, 2,16,14,22,17,20, 4, 9,10,11,15, 8,12,19,18,13, 7,21},
  { 3,16, 5,11,21, 9, 2,15,10,18,17, 7, 8,19,14,13, 1, 4,22,20, 6,12},
  {14, 4,13, 5,21,11, 8, 6, 3,17, 2,20, 1,19,10, 9,12,18,15, 7,22,16},
  { 4,12,22,14, 5,15,16, 2, 8,10,17, 9,21, 7, 3, 6,13,18,11,20,19, 1},
  {16,15,14,13, 4,22,18,19, 7, 1,12,17, 5,10,20, 3, 9,21,11, 2, 6, 8},
  {20,14,21,12,15,17, 4,19,13,10,11, 1,16, 5,18, 7, 8,22, 9, 6, 3, 2},
  {16,14,13, 2,21,10,11, 4, 1,22,18,12,19, 5, 7, 8, 6, 3,15,20, 9,17},
  {18,15, 9,14,12, 2, 8,11,22,21,16, 1, 6,17, 5,10,19, 4,20,13, 3, 7},
  { 7, 3,10,14,13,21,18, 6,20, 4, 9, 8,22,15, 2, 1, 5,12,19,17,11,16},
  {18, 1,13, 7,16,10,14, 2,19, 5,21,11,22,15, 8,17,20, 3, 4,12, 6, 9},
  {13, 2,22, 5,11,21,20,14, 7,10, 4, 9,19,18, 6, 3, 1, 8,15,12,17,16},
  {14,17,21, 8, 2, 9, 6, 4, 5,13,22, 7,15, 3, 1,18,16,11,10,12,20,19},
  {10,22, 1,12,13,18,21,20, 2,14,16, 7,15, 3, 4,17, 5,19, 6, 8, 9,11},
  {10, 8, 9,18,12, 6, 1, 5,20,11,17,22,16, 3,13, 2,15,21,14,19, 7, 4},
  { 7,17,22, 5, 3,10,13,18, 9, 1,14,15,21,19,16,12, 8, 6,11,20, 4, 2},
  { 2, 9,21, 3, 4, 7, 1,11,16, 5,20,19,18, 8,17,13,10,12,15, 6,14,22},
  {15,12, 8, 4,22,13,16,17,18, 3, 7, 5, 6, 1, 9,11,21,10,14,20,19, 2},
  {15,16, 2,11,17, 7, 5,14,20, 4,21, 3,10, 9,12, 8,13, 6,18,19,22, 1},
  { 1,13,11, 3, 4,21, 6,14,15,22,18, 9, 7, 5,10,20,12,16,17, 8,19, 2},
  {14,17,22,20, 8,16, 5,10, 1,13, 2,21,12, 9, 4,18, 3, 7, 6,19,15,11},
  { 9,17, 7, 4, 5,13,21,18,11, 3,22, 1, 6,16,20,14,15,10, 8, 2,12,19},
  {13,14, 5,22,19,11, 9, 6,18,15, 8,10, 7, 4,17,16, 3, 1,12, 2,21,20},
  {20, 5, 4,14,11, 1, 6,16, 8,22, 7, 3, 2,12,21,19,17,13,10,15,18, 9},
  { 3, 7,14,15, 6, 5,21,20,18,10, 4,16,19, 1,13, 9, 8,17,11,12,22, 2},
  {13,15,17, 1,22,11, 3, 4, 7,20,14,21, 9, 8, 2,18,16, 6,10,12, 5,19}
};
```

print.c

```
#ifdef RCSID
static char *RCSid =
   "$Header: print.c 11-apr-2001.14:53:52 mpoess Exp $ ";
#endif /* RCSID */

/* Copyright (c) Oracle Corporation 1999, 2000. All Rights Reserved. */

/*

   NAME
     print.c - <one-line expansion of the name>

   DESCRIPTION
     <short description of component this file declares/defines>

   PUBLIC FUNCTION(S)
     <list of external functions declared/defined - with one-line
descriptions>

   PRIVATE FUNCTION(S)
     <list of static functions defined in .c file - with one-line
descriptions>

   RETURNS
     <function return values, for .c file with single function>

   NOTES
     <other useful comments, qualifications, etc.>

   MODIFIED   (MM/DD/YY)
   mpoess     04/11/01 -
   mpoess     02/22/01 - add linux support
   mpoess     08/28/00 - add support for SF>1000
   mpoess     06/14/00 - change order of partsupp fields to be
partkey,suppkey,supplycost,availqty,comment
   mpoess     07/12/99 - exchange with new version July 8 1999
   mpoess     06/28/99 - Creation
   mpoess     06/28/99 - Creation

*/


/* @(#)print.c            2.1.8.2 */
/* generate flat files for data load */
#include <stdio.h>
#ifndef VMS
#include <sys/types.h>
#endif
#if defined(SUN)
#include <unistd.h>
#endif
#include <math.h>

#include "dss.h"
#include "dsstypes.h"
#include <string.h>

#ifdef ORACLE
#include "part.h"

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

int slen;                       /* current string length        */
int wlen;                       /* number of bytes written      */

#ifdef DEBUG
long row_count;                 /* total number of rows generated     */
long *part_row;                 /* number of rows per partition       */
#endif /* DEBUG */

extern void Exitprog();

#endif /* ORACLE */

/*
 * 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 */
                        if ( insert_segments )
                                {
                                int this_segment;

        if(strcmp(tdefs[table].name,"orders.tbl"))

        this_segment=++insert_orders_segment;
                                else

        this_segment=++insert_lineitem_segment;
                                sprintf(upath,
"%s%c%s.u%d.%d",

        env_config(PATH_TAG, PATH_DFLT),
                                                PATH_SEP,
tdefs[table].name, update%10000,this_segment);
                                }
                        else
                                {
                                sprintf(upath,
"%s%c%s.u%d",
                                env_config(PATH_TAG,
PATH_DFLT),
                                PATH_SEP,
tdefs[table].name, update);
                                }
                else /* deletes */
                        if ( delete_segments )
                                {
                                ++delete_segment;
                                sprintf(upath,
"%s%cdelete.u%d.%d",

        env_config(PATH_TAG, PATH_DFLT), PATH_SEP, -
update%10000,

        delete_segment);
                                }
                        else
                                {
                                sprintf(upath,
"%s%cdelete.%d",
                                env_config(PATH_TAG,
PATH_DFLT), PATH_SEP, -update);
                                }
                return(fopen(upath, "w"));
        }
    res = tbl_open(table, "w");
    OPEN_CHECK(res, tdefs[table].name);
    return(res);
}

int
dbg_print(int format, FILE *tgt, void *data, int len, int sep)
{
        int dollars,
                cents;

        switch(format)
        {
        case DT_STR:
                if (columnar)
                        fprintf(tgt, "%-*s", len, (char *)data);
                else
                        fprintf(tgt, "%s", (char *)data);
                break;
#ifdef MVS
        case DT_VSTR:
                /* note: only used in MVS, assumes columnar
output */
                fprintf(tgt, "%c%c%-*s",
                        (len >> 8) & 0xFF, len & 0xFF, len,
(char *)data);
                break;
#endif /* MVS */
        case DT_INT:
                if (columnar)
                        fprintf(tgt, "%12ld", (long)data);
                else
                        fprintf(tgt, "%ld", (long)data);
                break;
        case DT_HUGE:
#ifndef SUPPORT_64BITS
      if (*(long *)((long *)data + 1) == 0) \
        if (columnar) fprintf(tgt, "%12ld", *(long *)data);
        else fprintf(tgt, "%ld", *(long *)data);
      else
        if (columnar) fprintf(tgt, "%5ld%07ld",
                                        *(long *)((long *)data + 1),
*(long *)data);
        else fprintf(tgt,"%ld%07ld",
                                        *(long *)((long *)data + 1),
*(long *)data);
#else
        /* printf ("before writing to disk: data is
%lld\n",*(DSS_HUGE *)data);*/

                fprintf(tgt, HUGE_FORMAT, *(DSS_HUGE
*)data);
#endif /* SUPPORT_64BITS */
                break;
        case DT_KEY:
                fprintf(tgt, "%ld", (long)data);
                break;
        case DT_MONEY:
                cents = (long)data;
                if (cents < 0)
                        {
                        fprintf(tgt, "-");
                        cents = -cents;
                        }
                dollars = cents / 100;
                cents %= 100;
                if (columnar)
                        fprintf(tgt, "%12ld.%02ld", dollars,
cents);
                else
                        fprintf(tgt, "%ld.%02ld", dollars,
cents);
                break;
        case DT_CHR:
                if (columnar)
#ifdef LINUX
                        fprintf(tgt, "%c ", *(char *)data);
#else
                        fprintf(tgt, "%c ", (char)data);
#endif
                else
#ifdef LINUX
                        fprintf(tgt, "%c", *(char *)data);
#else
                        fprintf(tgt, "%c", (char)data);
#endif
                break;
        }

#ifdef EOL_HANDLING
        if (sep)
#endif /* EOL_HANDLING */
```

```
        if (!columnar && (sep != -1))
                fprintf(tgt, "%c", SEPARATOR);

        return(0);
}

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

#ifdef ORACLE
   if (partition) {
      partnum = find_partition_cust(c);
      fp = out[partnum];
   } else if (fp == NULL)
       fp = print_prep(CUST, 0);

  PR_STRT(fp);
  PR_INT(fp, c->custkey);
  PR_STR(fp, c->mktsegment, C_MSEG_LEN);
  PR_INT(fp, c->nation_code);
  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_STR(fp, c->phone, PHONE_LEN);
  PR_MONEY(fp, c->acctbal);
  PR_VSTR_LAST(fp, c->comment,
     (columnar)?(long)(ceil(C_CMNT_LEN * V_STR_HGH)):c->clen);
  PR_END(fp);
#else /* !ORACLE */
  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_LAST(fp, c->comment,
     (columnar)?(long)(ceil(C_CMNT_LEN * V_STR_HGH)):c->clen);
  PR_END(fp);
#endif /* !ORACLE */

  return(0);
}

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

#ifdef ORACLE
   if (partition)
   {
      partnum = find_partition_order(o);
      fp_o = out[partnum];
      if (mode != last_mode)
         last_mode = mode;
   } else
   {
      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_STR(fp_o, o->odate, DATE_LEN);
   /* printf ("o->okey=%lld\n",*o->okey); */
   PR_HUGE(fp_o, o->okey);
   PR_INT(fp_o, o->custkey);
   PR_STR(fp_o, o->opriority, O_OPRIO_LEN);
   PR_INT(fp_o, o->spriority);
   PR_STR(fp_o, o->clerk, O_CLRK_LEN);
#ifdef LINUX
   PR_CHR(fp_o, &(o->orderstatus));
#else
   PR_CHR(fp_o, o->orderstatus);
#endif
   PR_MONEY(fp_o, o->totalprice);
   PR_VSTR_LAST(fp_o, o->comment,
     (columnar)?(long)(ceil(O_CMNT_LEN * V_STR_HGH)):o->clen);
   PR_END(fp_o);
#else /* !ORACLE */
   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_HUGE(fp_o, o->okey);
   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_OPRIO_LEN);
   PR_STR(fp_o, o->clerk, O_CLRK_LEN);
   PR_INT(fp_o, o->spriority);
     (columnar)?(long)(ceil(O_CMNT_LEN * V_STR_HGH)):o->clen);
   PR_END(fp_o);
#endif /* !ORACLE */

   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;

#ifdef ORACLE
   if (!partition)
   {
      if (fp_l == NULL || mode != last_mode)
      {
         if (fp_l)
            fclose(fp_l);
         fp_l = print_prep(LINE, mode);
         last_mode = mode;
      }
   } else {
      if ( mode != last_mode)
         last_mode = mode;
   }

   for (i = 0; i < o->lines; i++)
   {
      if (partition) {
```

```
            partnum = find_partition_line(o, i);
            fp_l = out[partnum];
        }
        PR_STRT(fp_l);
        PR_STR(fp_l, o->l[i].sdate, DATE_LEN);
        PR_HUGE(fp_l, o->okey);
        PR_MONEY(fp_l, o->l[i].discount);
        PR_MONEY(fp_l, o->l[i].eprice);
        PR_INT(fp_l, o->l[i].suppkey);
        PR_INT(fp_l, o->l[i].quantity);
#ifdef LINUX
        PR_CHR(fp_l, &(o->l[i].rflag[0]));
#else
        PR_CHR(fp_l, o->l[i].rflag[0]);
#endif
        PR_INT(fp_l, o->l[i].partkey);
#ifdef LINUX
        PR_CHR(fp_l, &(o->l[i].lstatus[0]));
#else
        PR_CHR(fp_l, o->l[i].lstatus[0]);
#endif
        PR_MONEY(fp_l, o->l[i].tax);
        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].shipmode, L_SMODE_LEN);
        PR_INT(fp_l, o->l[i].lcnt);
        PR_STR(fp_l, o->l[i].shipinstruct, L_INST_LEN);
        PR_VSTR_LAST(fp_l, o->l[i].comment,
            (columnar)?(long)(ceil(L_CMNT_LEN * V_STR_HGH)):o-
>l[i].clen);
        PR_END(fp_l);
    }
#else /* !ORACLE */
    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_HUGE(fp_l, o->l[i].okey);
        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_LAST(fp_l, o->l[i].comment,
            (columnar)?(long)(ceil(L_CMNT_LEN * V_STR_HGH)):o-
>l[i].clen);
        PR_END(fp_l);
        }
#endif /* !ORACLE */

    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;

#ifdef ORACLE

    if (partition)
    {
        partnum = find_partition_part(part);
        p_fp = out[partnum];
    } else
    {
    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->type,
        (columnar)?(long)P_TYPE_LEN:part->tlen);
    PR_INT(p_fp, part->size);
    PR_STR(p_fp, part->brand, P_BRND_LEN);
    PR_VSTR(p_fp, part->name,
        (columnar)?(long)P_NAME_LEN:part->nlen);
    PR_STR(p_fp, part->container, P_CNTR_LEN);
    PR_STR(p_fp, part->mfgr, P_MFG_LEN);
    PR_MONEY(p_fp, part->retailprice);
    PR_VSTR_LAST(p_fp, part->comment,
        (columnar)?(long)(ceil(P_CMNT_LEN * V_STR_HGH)):part-
>clen);
    PR_END(p_fp);
#else /* !ORACLE */
     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->mfgr, P_MFG_LEN);
    PR_STR(p_fp, part->brand, 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_LAST(p_fp, part->comment,
        (columnar)?(long)(ceil(P_CMNT_LEN * V_STR_HGH)):part-
>clen);
    PR_END(p_fp);
#endif /* !ORACLE */

    return(0);
}

/*
```

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

#ifdef ORACLE
   if ((!partition) && (ps_fp == NULL))
      ps_fp = print_prep(PSUPP, mode);
#else /* !ORACLE */
   if (ps_fp == NULL)
      ps_fp = print_prep(PSUPP, mode);
#endif /* !ORACLE */

  for (i = 0; i < SUPP_PER_PART; i++)
     {
#ifdef ORACLE
     if (partition)
     {
        partnum = find_partition_psupp(part);
        ps_fp = out[partnum];
     }
#endif /* ORACLE */
     PR_STRT(ps_fp);
     PR_INT(ps_fp, part->s[i].partkey);
     PR_INT(ps_fp, part->s[i].suppkey);
#ifdef ORACLE
     PR_MONEY(ps_fp, part->s[i].scost);
     PR_INT(ps_fp, part->s[i].qty);
#endif /* ORACLE */
#ifndef ORACLE
     PR_INT(ps_fp, part->s[i].qty);
     PR_MONEY(ps_fp, part->s[i].scost);
#endif /* !ORACLE */
     PR_VSTR_LAST(ps_fp, part->s[i].comment,
      (columnar)?(long)(ceil(PS_CMNT_LEN * V_STR_HGH)):part-
>s[i].clen);
     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;

#ifdef ORACLE
  if (partition)
  {
     partnum = find_partition_supp(supp);
     fp = out[partnum];
  } else if (fp == NULL)
      fp = print_prep(SUPP, mode);
  PR_STRT(fp);
  PR_INT(fp, supp->suppkey);
  PR_INT(fp, supp->nation_code);
  PR_VSTR(fp, supp->comment,
     (columnar)?(long)(ceil(S_CMNT_LEN * V_STR_HGH)):supp-
>clen);
  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_STR(fp, supp->phone, PHONE_LEN);
  PR_MONEY(fp, supp->acctbal);
  PR_END(fp);
#else /* !ORACLE */
  if (fp == NULL)
     fp = print_prep(SUPP, mode);
  PR_STRT(fp);
  PR_INT(fp, supp->suppkey);
  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_LAST(fp, supp->comment,
     (columnar)?(long)(ceil(S_CMNT_LEN * V_STR_HGH)):supp-
>clen);
  PR_END(fp);
#endif /* !ORACLE */

  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_LAST(fp, c->comment,
     (columnar)?(long)(ceil(N_CMNT_LEN * V_STR_HGH)):c->clen);
  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_LAST(fp, c->comment,
     (columnar)?(long)(ceil(R_CMNT_LEN * V_STR_HGH)):c->clen);
  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;

            static int rows_per_segment=0;
            static int rows_this_segment=0;
      static int residual_rows=0;

   if (last_num != num)
      {
      if (dfp)
         fclose(dfp);
      dfp = print_prep(tbl, -num);
      if (dfp == NULL)
         return(-1);
      last_num = num;
                        rows_this_segment=0;
      }

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

            if(rows_per_segment==0)
                                                                  {

            rows_per_segment = (cnt / delete_segments);

            residual_rows = (cnt % delete_segments);

            rows_per_segment++;
                                                                  }

            if(delete_segment <= residual_rows)
                                                                  {
            if((++rows_this_segment) > rows_per_segment)

            {

            fclose(dfp);

            dfp = print_prep(tbl, -num);

            if (dfp == NULL) return(-1);

            last_num = num;

            rows_this_segment=1;

            }
                                                                        }
                                                                  else
                                                                        {

            if((++rows_this_segment) >= rows_per_segment)

            {

            fclose(dfp);

            dfp = print_prep(tbl, -num);

            if (dfp == NULL) return(-1);

            last_num = num;

            rows_this_segment=1;

            }
                                                                        }
                  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);
}

/*
 * verify functions: routines which replace the pr_routines and generate
a pseudo checksum
 * instead of generating the actual contents of the tables. Meant to allow
large scale data
 * validation without requiring a large amount of storage
 */
int
vrf_cust(customer_t *c, int mode)
{
   VRF_STRT(CUST);
   VRF_INT(CUST, c->custkey);
   VRF_STR(CUST, c->name);
   VRF_STR(CUST, c->address);
   VRF_INT(CUST, c->nation_code);
```

```
  VRF_STR(CUST, c->phone);
  VRF_MONEY(CUST, c->acctbal);
  VRF_STR(CUST, c->mktsegment);
  VRF_STR(CUST, c->comment);
  VRF_END(CUST);

  return(0);
}

/*
 * print the numbered order
 */
int
vrf_order(order_t *o, int mode)
{
   VRF_STRT(ORDER);
   VRF_HUGE(ORDER, o->okey);
   VRF_INT(ORDER, o->custkey);
   VRF_CHR(ORDER, o->orderstatus);
   VRF_MONEY(ORDER, o->totalprice);
   VRF_STR(ORDER, o->odate);
   VRF_STR(ORDER, o->opriority);
   VRF_STR(ORDER, o->clerk);
   VRF_INT(ORDER, o->spriority);
   VRF_STR(ORDER, o->comment);
   VRF_END(ORDER);

   return(0);
}

/*
 * print an order's lineitems
 */
int
vrf_line(order_t *o, int mode)
{
                int i;

   for (i = 0; i < o->lines; i++)
      {
      VRF_STRT(LINE);
      VRF_HUGE(LINE, o->l[i].okey);
      VRF_INT(LINE, o->l[i].partkey);
      VRF_INT(LINE, o->l[i].suppkey);
      VRF_INT(LINE, o->l[i].lcnt);
      VRF_INT(LINE, o->l[i].quantity);
      VRF_MONEY(LINE, o->l[i].eprice);
      VRF_MONEY(LINE, o->l[i].discount);
      VRF_MONEY(LINE, o->l[i].tax);
      VRF_CHR(LINE, o->l[i].rflag[0]);
      VRF_CHR(LINE, o->l[i].lstatus[0]);
      VRF_STR(LINE, o->l[i].sdate);
      VRF_STR(LINE, o->l[i].cdate);
      VRF_STR(LINE, o->l[i].rdate);
      VRF_STR(LINE, o->l[i].shipinstruct);
      VRF_STR(LINE, o->l[i].shipmode);
      VRF_STR(LINE, o->l[i].comment);
      VRF_END(LINE);
      }

  return(0);
}

/*
 * print the numbered order *and* its associated lineitems
 */
int
vrf_order_line(order_t *o, int mode)
{
   vrf_order(o, mode);
   vrf_line(o, mode);

   return(0);
}

/*
 * print the given part
 */
int
vrf_part(part_t *part, int mode)
{

  VRF_STRT(PART);
  VRF_INT(PART, part->partkey);
  VRF_STR(PART, part->name);
  VRF_STR(PART, part->mfgr);
  VRF_STR(PART, part->brand);
  VRF_STR(PART, part->type);
  VRF_INT(PART, part->size);
  VRF_STR(PART, part->container);
  VRF_MONEY(PART, part->retailprice);
  VRF_STR(PART, part->comment);
  VRF_END(PART);

  return(0);
}

/*
 * print the given part's suppliers
 */
int
vrf_psupp(part_t *part, int mode)
{
   long     i;

  for (i = 0; i < SUPP_PER_PART; i++)
     {
     VRF_STRT(PSUPP);
     VRF_INT(PSUPP, part->s[i].partkey);
     VRF_INT(PSUPP, part->s[i].suppkey);
     VRF_INT(PSUPP, part->s[i].qty);
     VRF_MONEY(PSUPP, part->s[i].scost);
     VRF_STR(PSUPP, part->s[i].comment);
     VRF_END(PSUPP);
     }

  return(0);
}

/*
 * print the given part *and* its suppliers
 */
int
vrf_part_psupp(part_t *part, int mode)
{
   vrf_part(part, mode);
   vrf_psupp(part, mode);

   return(0);
}

int
vrf_supp(supplier_t *supp, int mode)
{
  VRF_STRT(SUPP);
  VRF_INT(SUPP, supp->suppkey);
  VRF_STR(SUPP, supp->name);
  VRF_STR(SUPP, supp->address);
  VRF_INT(SUPP, supp->nation_code);
  VRF_STR(SUPP, supp->phone);
  VRF_MONEY(SUPP, supp->acctbal);
  VRF_STR(SUPP, supp->comment);
  VRF_END(SUPP);

  return(0);
}
```

```
int
vrf_nation(code_t *c, int mode)
{
  VRF_STRT(NATION);
  VRF_INT(NATION, c->code);
  VRF_STR(NATION, c->text);
  VRF_INT(NATION, c->join);
  VRF_STR(NATION, c->comment);
  VRF_END(NATION);

  return(0);
}

int
vrf_region(code_t *c, int mode)
{
  VRF_STRT(REGION);
  VRF_INT(REGION, c->code);
  VRF_STR(REGION, c->text);
  VRF_STR(REGION, c->comment);
  VRF_END(fp);

  return(0);
}
```

qgen.c

```
#ifdef RCSID
static char *RCSid =
  "$Header: qgen.c 06-feb-2001.20:00:57 mpoess Exp $ ";
#endif /* RCSID */

/* Copyright (c) Oracle Corporation 1999, 2000. All Rights Reserved. */

/*

  NAME
    qgen.c - <one-line expansion of the name>

  DESCRIPTION
    <short description of component this file declares/defines>

  PUBLIC FUNCTION(S)
    <list of external functions declared/defined - with one-line
descriptions>

  PRIVATE FUNCTION(S)
    <list of static functions defined in .c file - with one-line
descriptions>

  RETURNS
    <function return values, for .c file with single function>

  NOTES
    <other useful comments, qualifications, etc.>

  MODIFIED   (MM/DD/YY)
  mpoess     02/06/01 - debug -q option
  mpoess     08/30/00 - add support for streamstamp
  mpoess     08/30/00 - add support for stream generation with subset of
queries
  mpoess     08/31/99 - add Jacks changes for version 1.1.0/1.2.0 (H/R)
as of
  mpoess     08/05/99 - add striping of comments starting with Rem
  mpoess     07/12/99 - exchange with new version July 8 1999
  mpoess     06/28/99 - Creation
  mpoess     06/28/99 - Creation

*/

/*
 * Sccsid:     @(#)qgen.c              2.1.8.2
 * qgen.c -- routines to convert query templates to executable query
 *           text for TPC-H and TPC-R
 */
#define DECLARER

#include <stdio.h>
#include <string.h>
#if (defined(_POSIX_)||!defined(WIN32))
#include <unistd.h>
#else
#include "process.h"
#endif /* WIN32 */
#include <ctype.h>
#include <time.h>
#include "config.h"
#include "dss.h"
#include "tpcd.h"
#include "permute.h"


#define LINE_SIZE 512

/*
 * Function Protoypes
 */
void varsub PROTO((int qnum, int vnum, int flags));
int strip_comments PROTO((char *line));
void usage PROTO((void));
int process_options PROTO((int cnt, char **args));
int setup PROTO((void));
void qsub PROTO((char *qtag, int flags));



extern char *optarg;
extern int optind;
char **mk_ascdate(void);
extern seed_t Seed[];

char **asc_date;
int snum = -1;
char *prog;
int  queryset= 4194303; /* this is a integer holding the bits of 22 queries
*/
tdef tdefs = { NULL };
long rndm;
double flt_scale;
distribution q13a, q13b;
int qnum;
int pmonitor=0;

/*
 * FUNCTION strip_comments(line)
 *
 * remove all comments from 'line'; recognizes both {} and -- comments
 */
int
strip_comments(char *line)
{
   static int in_comment = 0;
   char *cp1, *cp2;

   cp1 = line;

   while (1)   /* traverse the entire string */
      {
      if (in_comment)
         {
         if ((cp2 = strchr(cp1, '}')) != NULL) /* comment ends */
```

```
            {
            strcpy(cp1, cp2 + 1);
            in_comment = 0;
            continue;
            }
         else
            {
            *cp1 = '\0';
            break;
            }
         }
      else    /* not in_comment */
         {
         if ((cp2 = strchr(cp1, '-')) != NULL)
            {
            if (*(cp2 + 1) == '-')  /* found a '--' comment */
               {
               *cp2 = '\0';
               break;
               }
            }
#ifndef ORIG
                if ((cp2 = strchr(cp1, 'R')) != NULL)
            {
            if (*(cp2 + 1) == 'e')
               {
               if (*(cp2 + 2) == 'm')
                  {
                  *cp2 = '\0';
                  break;
                  }
               }
            }
#endif
         if ((cp2 = strchr(cp1, '{')) != NULL) /* comment starts */
            {
            in_comment = 1;
            *cp2 = ' ';
            continue;
            }
         else break;
         }
      }
   return(0);
}

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

   qnum = atoi(qtag);
   if (line == NULL)
      {
      line = malloc(BUFSIZ);
      qpath = malloc(BUFSIZ);
      MALLOC_CHECK(line);
      MALLOC_CHECK(qpath);
      }

   qroot = env_config(QDIR_TAG, QDIR_DFLT);
   sprintf(qpath, "%s%c%s.sql",
                           qroot, PATH_SEP, qtag);
   qfp = fopen(qpath, "r");
   OPEN_CHECK(qfp, qpath);

   rowcnt = rowcnt_dflt[qnum];
   varsub(qnum, 0, flags); /* set the variables */
   if (flags & DFLT_NUM)
      fprintf(ofp, SET_ROWCOUNT, rowcnt);
   while (fgets(line, BUFSIZ, qfp) != NULL)
      {
      if (!(flags & COMMENT))
         strip_comments(line);
      mark = line;
      while ((cptr = strchr(mark, VTAG)) != NULL)
         {
         *cptr = '\0';
          cptr++;
         fprintf(ofp,"%s", mark);
         switch(*cptr)
            {
            case 'b':
            case 'B':
               if (!(flags & ANSI))
                  fprintf(ofp,"%s\n", START_TRAN);
               cptr++;
               break;
            case 'c':
            case 'C':
               if (flags & DBASE)
                  fprintf(ofp, SET_DBASE, db_name);
               cptr++;
               break;
            case 'e':
            case 'E':
               if (!(flags & ANSI))
                  fprintf(ofp,"%s\n", END_TRAN);
               cptr++;
               break;
            case 'n':
            case 'N':
               if (!(flags & DFLT_NUM))
                  {
                  rowcnt=atoi(++cptr);
                  while (isdigit(*cptr) || *cptr == ' ') cptr++;
                  fprintf(ofp, SET_ROWCOUNT, rowcnt);
                  }
               continue;
            case 'o':
            case 'O':
               if (flags & OUTPUT)
                  fprintf(ofp,"%s '%s/%s.%d'", SET_OUTPUT, osuff,
                     qtag, (snum < 0)?0:snum);
               cptr++;
               break;
            case 'q':
            case 'Q':
               fprintf(ofp,"%s", qtag);
```

```
            cptr++;
            break;
         case 's':
         case 'S':
            fprintf(ofp,"%d", (snum < 0)?0:snum);
            cptr++;
            break;
         case 'X':
         case 'x':
            if (flags & EXPLAIN)
               fprintf(ofp, "%s\n", GEN_QUERY_PLAN);
            cptr++;
            break;
                  case '1':
                  case '2':
                  case '3':
                  case '4':
                  case '5':
                  case '6':
                  case '7':
                  case '8':
                  case '9':
            varsub(qnum, atoi(cptr), flags & DFLT);
            while (isdigit(*++cptr));
            break;
         default:
                     fprintf(stderr, "-- unknown flag '%c%c'
ignored\n",
               VTAG, *cptr);
                     cptr++;
                     break;
         }
       mark=cptr;
       }
     fprintf(ofp,"%s", mark);
     }
  fclose(qfp);
  fflush(stdout);
  return;
}

void
usage(void)
{
printf("%s Parameter Substitution (v. %d.%d.%d%s)\n",
       NAME, VERSION,RELEASE,
        MODIFICATION,PATCH);
printf("Copyright %s %s\n", TPC, C_DATES);
printf("USAGE: %s <options> [ queries ]\n", prog);
printf("Options:\n");
printf("\t-a\t\t-- use ANSI semantics.\n");
printf("\t-b <str>\t-- load distributions from <str>\n");
printf("\t-c\t\t-- retain comments found in template.\n");
printf("\t-d\t\t-- use default substitution values.\n");
printf("\t-h\t\t-- print this usage summary.\n");
printf("\t-i <str>\t-- use the contents of file <str> to begin a query.\n");
printf("\t-l <str>\t-- log parameters to <str>.\n");
printf("\t-n <str>\t-- connect to database <str>.\n");
printf("\t-N\t\t-- use default rowcounts and ignore :n directive.\n");
printf("\t-o <str>\t-- set the output file base path to <str>.\n");
printf("\t-p <n>\t\t-- use the query permutation for stream <n>\n");
printf("\t-r <n>\t\t-- seed the random number generator with <n>\n");
printf("\t-s <n>\t\t-- base substitutions on an SF of <n>\n");
printf("\t-v\t\t-- verbose.\n");
printf("\t-t <str>\t-- use the contents of file <str> to complete a
query\n");
printf("\t-x\t\t-- enable SET EXPLAIN in each query.\n");
printf("\t-q\t\t-- list of queries to include in the query set, separated by
comma.\n");
}

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

   while((flag = getopt(cnt, args, "ab:cdhi:n:m:Nl:o:p:q:r:s:t:vx")) != -1)
      switch(flag)
         {
         case 'a':   /* use ANSI semantics */
            flags |= ANSI;
            break;
                                    case 'b':            /* load distributions
from named file */
                                       d_path = (char
*)malloc(strlen(optarg) + 1);

               MALLOC_CHECK(d_path);
                                       strcpy(d_path, optarg);
                                       break;
                                    case 'c':   /* retain comments in EQT
*/
            flags |= COMMENT;
            break;
         case 'd':   /* use default substitution values */
            flags |= DFLT;
            break;
         case 'h':   /* just generate the usage summary */
            usage();
            exit(0);
            break;
         case 'i':   /* set stream initialization file name */
            ifile = malloc(strlen(optarg) + 1);
            MALLOC_CHECK(ifile);
            strcpy(ifile, optarg);
            flags |= INIT;
            break;
         case 'l':   /* log parameter usages */
            lfile = malloc(strlen(optarg) + 1);
            MALLOC_CHECK(lfile);
            strcpy(lfile, optarg);
            flags |= LOG;
            break;
         case 'm':   /* monitor performance */
            pmonitor = 1;
            break;
         case 'N':   /* use default rowcounts */
            flags |= DFLT_NUM;
            break;
         case 'n':   /* set database name */
            db_name = malloc(strlen(optarg) + 1);
            MALLOC_CHECK(db_name);
            strcpy(db_name, optarg);
            flags |= DBASE;
            break;
         case 'o':   /* set the output path */
            osuff = malloc(strlen(optarg) + 1);
            MALLOC_CHECK(osuff);
            strcpy(osuff, optarg);
            flags |=OUTPUT;
            break;
         case 'p':   /* permutation for a given stream */
            snum = atoi(optarg);
            break;
         case 'q':   /* permutation for a given stream */
                     queryset = 0;
                     for (i=0;i<strlen(optarg);i++) {
                       if (optarg[i] != ',') {
                         queryset = (queryset | ( 1 <<  ( ( optarg[i] - 48 )
- 1) ) );
                       }
                     }
```

```
            break;
        case 'r':   /* set random number seed for parameter gen */
            flags |= SEED;
            rndm = atol(optarg);
            break;
        case 's':   /* scale of data set to run against */
            flt_scale = atof(optarg);
                                                if (scale > MAX_SCALE)
                                                        fprintf(stderr,
"%s %5.0f %s\n%s\n",

                "WARNING: Support for scale factors >",

                MAX_SCALE,
                                                                "GB
is still in development.",

                "Data set integrity is not guaranteed.\n");
            break;
        case 't':   /* set termination file name */
            tfile = malloc(strlen(optarg) + 1);
            MALLOC_CHECK(tfile);
            strcpy(tfile, optarg);
            flags |= TERMINATE;
            break;
        case 'v':   /* verbose */
            flags |= VERBOSE;
            break;
        case 'x':   /* set explain in the queries */
            flags |= EXPLAIN;
            break;
        default:
            printf("unknown option '%s' ignored\n", args[optind]);
            usage();
            exit(1);
            break;
        }
    return(0);
}

int
setup(void)
{
    asc_date = mk_ascdate();
    read_dist(env_config(DIST_TAG, DIST_DFLT), "p_cntr",
&p_cntr_set);
    read_dist(env_config(DIST_TAG, DIST_DFLT), "colors", &colors);
    read_dist(env_config(DIST_TAG, DIST_DFLT), "p_types",
&p_types_set);
    read_dist(env_config(DIST_TAG, DIST_DFLT), "nations",
&nations);
    read_dist(env_config(DIST_TAG, DIST_DFLT), "nations2",
&nations2);
    read_dist(env_config(DIST_TAG, DIST_DFLT), "regions",
&regions);
    read_dist(env_config(DIST_TAG, DIST_DFLT), "o_oprio",
       &o_priority_set);
    read_dist(env_config(DIST_TAG, DIST_DFLT), "instruct",
       &l_instruct_set);
    read_dist(env_config(DIST_TAG, DIST_DFLT), "smode",
&l_smode_set);
    read_dist(env_config(DIST_TAG, DIST_DFLT), "category",
       &l_category_set);
    read_dist(env_config(DIST_TAG, DIST_DFLT), "rflag",
&l_rflag_set);
    read_dist(env_config(DIST_TAG, DIST_DFLT), "msegmnt",
&c_mseg_set);
                read_dist(env_config(DIST_TAG, DIST_DFLT), "Q13a",
&q13a);
                read_dist(env_config(DIST_TAG, DIST_DFLT), "Q13b",
&q13b);

    return(0);
}


main(int ac, char **av)
{
    int i;
    FILE *ifp;
    char line[LINE_SIZE];

    prog = av[0];
    flt_scale = (double)1.0;
    flags = 0;
                d_path = NULL;
    process_options(ac, av);
    if (flags & VERBOSE)
        fprintf(ofp,
                "-- TPC %s Parameter Substitution (Version
%d.%d.%d%s)\n",
            NAME, VERSION, RELEASE, MODIFICATION, PATCH);

    setup();

    if (!(flags & DFLT))        /* perturb the RNG */
                {
                if (!(flags & SEED))
                rndm = (long)((unsigned)time(NULL) * DSS_PROC);
                        if (rndm < 0)
                                rndm += 2147483647;
                        Seed[0].value = rndm;
                        for (i=1; i <= QUERIES_PER_SET; i++)
                                {
                                Seed[0].value =
NextRand(Seed[0].value);
                                Seed[i].value = Seed[0].value;
                                }
                        printf("-- using %ld as a seed to the RNG\n",
rndm);
                    }
    else
        printf("-- using default substitutions\n");

    if (flags & INIT)           /* init stream with ifile */
        {
        ifp = fopen(ifile, "r");
              OPEN_CHECK(ifp, ifile);
        while (fgets(line, LINE_SIZE, ifp) != NULL)
            fprintf(stdout, "%s", line);
        }

    if (snum >= 0)
        if (optind < ac)
            for (i=optind; i < ac; i++)
                {
                char qname[10];
                sprintf(qname, "%d", SEQUENCE(snum, atoi(av[i])));
                qsub(qname, flags);
                }
        else
            for (i=1; i <= QUERIES_PER_SET; i++)
                {
                char qname[10];
                        /* printf ("I am about to generate the %d th query
of stream %d, query %d\n",i,snum,SEQUENCE(snum, i)); */
                        if ( (1 << (SEQUENCE(snum, i) - 1)) & queryset)
{
                          sprintf(qname, "%d", SEQUENCE(snum, i));
                          if (pmonitor == 1)
                            printf ("host stream_stamp Q%d start
\"$START_DATE\" \"$NUM_STREAM\" \"$LOG_FILE\"
\"$STAT_DIR\"\n",SEQUENCE(snum, i));
                          /* printf ("connect $FRAME_USER\n");*/
                          qsub(qname, flags);
                          if (pmonitor == 1) {
```

```
                        printf ("select \'#Q%d_degree# \'||
decode(LAST_QUERY, 0, 1, LAST_QUERY)degree from
v\\$pq_sesstat where STATISTIC=\'Server
Threads\';\n",SEQUENCE(snum, i));
                        printf ("select \'#Q%d_sid# \'|| sid,
\'#Q%d_serial# \' || serial# from v\\$session where audsid =
userenv(\'sessionid\');\n",SEQUENCE(snum, i),SEQUENCE(snum, i));
                        printf ("host stream_stamp Q%d end
\"$START_DATE\" \"$NUM_STREAM\" \"$LOG_FILE\"
\"$STAT_DIR\"\n",SEQUENCE(snum, i));
                      }
                    }
                    }
   else
      if (optind < ac)
         for (i=optind; i < ac; i++)
            qsub(av[i], flags);
      else
         for (i=1; i <= QUERIES_PER_SET; i++)
            {
            char qname[10];
            sprintf(qname, "%d", i);
            qsub(qname, flags);
            }

   if (flags & TERMINATE)      /* terminate stream with tfile */
      {
      ifp = fopen(tfile, "r");
      if (ifp == NULL)
            OPEN_CHECK(ifp, tfile);
      while (fgets(line, LINE_SIZE, ifp) != NULL)
         fprintf(stdout, "%s", line);
      }

   return(0);
}
```

rnd.c

```
#ifdef RCSID
static char *RCSid =
  "$Header: rnd.c 11-apr-2001.14:45:56 mpoess Exp $ ";
#endif /* RCSID */

/* Copyright (c) Oracle Corporation 1999. All Rights Reserved. */

/*

  NAME
    rnd.c - <one-line expansion of the name>

  DESCRIPTION
    <short description of component this file declares/defines>

  PUBLIC FUNCTION(S)
    <list of external functions declared/defined - with one-line
descriptions>

  PRIVATE FUNCTION(S)
    <list of static functions defined in .c file - with one-line
descriptions>

  RETURNS
    <function return values, for .c file with single function>

  NOTES
    <other useful comments, qualifications, etc.>

  MODIFIED   (MM/DD/YY)
  mpoess      04/11/01 -
  mpoess      07/12/99 - exchange with new version July 8 1999
  mpoess      06/28/99 - Creation

*/


/* @(#)rnd.c              2.1.8.1
 *
 *
 * RANDOM.C -- Implements Park & Miller's "Minimum Standard"
RNG
 *
 * (Reference:  CACM, Oct 1988, pp 1192-1201)
 *
 * NextRand:  Computes next random integer
 * UnifInt:   Yields an long uniformly distributed between given bounds
 * UnifReal: ields a real uniformly distributed between given bounds
 * Exponential: Yields a real exponentially distributed with given mean
 *
 */

#include "config.h"
#include <stdio.h>
#include <math.h>
#include "dss.h"
#include "rnd.h"

char *env_config PROTO((char *tag, char *dflt));
void NthElement(long, long *);

void
dss_random(long *tgt, long lower, long upper, long stream)
{
        *tgt = UnifInt((long)lower, (long)upper, (long)stream);
        Seed[stream].usage += 1;

        return;
}

void
row_start(int t)        \
{
        int i;
        for (i=0; i <= MAX_STREAM; i++)
                Seed[i].usage = 0 ;

        return;
}

void
row_stop(int t)         \
        {
        int i;

        /* need to allow for handling the master and detail together
*/
        if (t == ORDER_LINE)
                t = ORDER;
        if (t == PART_PSUPP)
                t = PART;

        for (i=0; i <= MAX_STREAM; i++)
                if ((Seed[i].table == t) || (Seed[i].table ==
tdefs[t].child))
                        {
                        if (set_seeds && (Seed[i].usage >
Seed[i].boundary))
                                {
                                fprintf(stderr, "\nSEED
CHANGE: seed[%d].usage = %d\n",
                                        i,
Seed[i].usage);
```

```
                                        Seed[i].boundary =
Seed[i].usage;
                                        }
                                else
                                        {

        NthElement((Seed[i].boundary - Seed[i].usage),
&Seed[i].value);
                                        }
                                }
                        return;
        }

void
dump_seeds(int tbl)
{
        int i;

        for (i=0; i <= MAX_STREAM; i++)
                if (Seed[i].table == tbl)
                        printf("%d:\t%ld\n", i, Seed[i].value);
        return;
}

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

  NextRand:  Computes next random integer

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

/*
 * long NextRand( long nSeed )
 */
long
NextRand(long nSeed)

/*
 * nSeed is the previous random number; the returned value is the
 * next random number. The routine generates all numbers in the
 * range 1 .. nM-1.
 */

{

   /*
    * The routine returns (nSeed * nA) mod nM, where   nA (the
    * multiplier) is 16807, and nM (the modulus) is
    * 2147483647 = 2^31 - 1.
    *
    * nM is prime and nA is a primitive element of the range 1..nM-1.
    * This * means that the map nSeed = (nSeed*nA) mod nM, starting
    * from any nSeed in 1..nM-1, runs through all elements of 1..nM-1
    * before repeating.  It never hits 0 or nM.
    *
    * To compute (nSeed * nA) mod nM without overflow, use the
    * following trick.  Write nM as nQ * nA + nR, where nQ = nM / nA
    * and nR = nM % nA.   (For nM = 2147483647 and nA = 16807,
    * get nQ = 127773 and nR = 2836.) Write nSeed as nU * nQ + nV,
    * where nU = nSeed / nQ and nV = nSeed % nQ.  Then we have:
    *
    * nM  =  nA * nQ  +  nR        nQ = nM / nA        nR < nA < nQ
    *
    * nSeed = nU * nQ  +  nV       nU = nSeed / nQ     nV < nU
    *
    * Since nA < nQ, we have nA*nQ < nM < nA*nQ + nA < nA*nQ +
nQ,
    * i.e., nM/nQ = nA.  This gives bounds on nU and nV as well:
    * nM > nSeed  =>  nM/nQ * >= nSeed/nQ  =>  nA >= nU ( > nV ).
    *
    * Using ~ to mean "congruent mod nM" this gives:
    *
    * nA * nSeed  ~  nA * (nU*nQ + nV)
    *
    * ~  nA*nU*nQ + nA*nV
    *
    * ~  nU * (-nR)  +  nA*nV      (as nA*nQ ~ -nR)
    *
    * Both products in the last sum can be computed without overflow
    * (i.e., both have absolute value < nM) since nU*nR < nA*nQ < nM,
    * and  nA*nV < nA*nQ < nM.  Since the two products have opposite
    * sign, their sum lies between -(nM-1) and +(nM-1).  If
    * non-negative, it is the answer (i.e., it's congruent to
    * nA*nSeed and lies between 0 and nM-1). Otherwise adding nM
    * yields a number still congruent to nA*nSeed, but now between
    * 0 and nM-1, so that's the answer.
    */

   long          nU, nV;

   nU = nSeed / nQ;
   nV = nSeed - nQ * nU;       /* i.e., nV = nSeed % nQ */
   nSeed = nA * nV - nU * nR;
   if (nSeed < 0)
      nSeed += nM;
   return (nSeed);
}

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

  UnifInt:  Yields an long uniformly distributed between given bounds

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

/*
 * long UnifInt( long nLow, long nHigh, long nStream )
 */
long
UnifInt(long nLow, long nHigh, long nStream)

/*
 * Returns an integer uniformly distributed between nLow and nHigh,
 * including * the endpoints.  nStream is the random number stream.
 * Stream 0 is used if nStream is not in the range 0..MAX_STREAM.
 */

{
   double         dRange;
   long           nTemp;

   if (nStream < 0 || nStream > MAX_STREAM)
      nStream = 0;

   if (nLow > nHigh)
   {
      nTemp = nLow;
      nLow = nHigh;
      nHigh = nTemp;
   }

   dRange = (double) (nHigh - nLow + 1);
   Seed[nStream].value = NextRand(Seed[nStream].value);
   nTemp = (long) (((double) Seed[nStream].value / dM) * (dRange));
   return (nLow + nTemp);
}



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

  UnifReal:  Yields a real uniformly distributed between given bounds
```

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

/*
 * double UnifReal( double dLow, double dHigh, long nStream )
 */
double
UnifReal(double dLow, double dHigh, long nStream)

/*
 * Returns a double uniformly distributed between dLow and dHigh,
 * excluding the endpoints.  nStream is the random number stream.
 * Stream 0 is used if nStream is not in the range 0..MAX_STREAM.
 */

{
   double         dTemp;

   if (nStream < 0 || nStream > MAX_STREAM)
      nStream = 0;
   if (dLow == dHigh)
      return (dLow);
   if (dLow > dHigh)
   {
      dTemp = dLow;
      dLow = dHigh;
      dHigh = dTemp;
   }
   Seed[nStream].value = NextRand(Seed[nStream].value);
   dTemp = ((double) Seed[nStream].value / dM) * (dHigh - dLow);
   return (dLow + dTemp);
}



/************************************************************
********%

  Exponential:  Yields a real exponentially distributed with given mean

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

/*
 * double Exponential( double dMean, long nStream )
 */
double
Exponential(double dMean, long nStream)

/*
 * Returns a double uniformly distributed with mean dMean.
 * 0.0 is returned iff dMean <= 0.0. nStream is the random number
 * stream. Stream 0 is used if nStream is not in the range
 * 0..MAX_STREAM.
 */

{
   double         dTemp;

   if (nStream < 0 || nStream > MAX_STREAM)
      nStream = 0;
   if (dMean <= 0.0)
      return (0.0);

   Seed[nStream].value = NextRand(Seed[nStream].value);
   dTemp = (double) Seed[nStream].value / dM;       /* unif between
0..1 */
   return (-dMean * log(1.0 - dTemp));
}
```

rnd.h

```
/*
 * $Header: rnd.h 11-apr-2001.14:45:03 mpoess Exp $
 */

/* Copyright (c) Oracle Corporation 1999. All Rights Reserved. */

/* NOTE:  See 'header_template.doc' in the 'doc' dve under the 'forms'
     directory for the header file template that includes instructions.
*/

/*
  NAME
    <<<LOWERCASE_FILENAME>>>.h - <one-line expansion of the
name>

  DESCRIPTION
    <short description of component this file declares/defines>

  RELATED DOCUMENTS

  INSPECTION STATUS
    Inspection date:
    Inspection status:
    Estimated increasing cost defects per page:
    Rule sets:

  ACCEPTANCE REVIEW STATUS
    Review date:
    Review status:
    Reviewers:

  PUBLIC FUNCTION(S)
    <list of external functions declared/defined - with one-line
descriptions>

  PRIVATE FUNCTION(S)
    <list of static functions defined in .c file - with one-line
descriptions>

  EXAMPLES

  NOTES
    <other useful comments, qualifications, etc.>

  MODIFIED   (MM/DD/YY)
  mpoess      04/11/01 -
  mpoess      07/12/99 - exchange with new version July 8 1999
  mpoess      06/28/99 - Creation
  mpoess      06/28/99 - Creation

/*
 * Sccsid:  @(#)rnd.h  2.1.8.1
 *
 * rnd.h -- header file for use withthe portable random number generator
 * provided by Frank Stephens of Unisys
 */

/* function protypes */
long           NextRand    PROTO((long));
long           UnifInt     PROTO((long, long, long));
double         UnifReal    PROTO((double, double, long));
double         Exponential PROTO((double, long));

static long     nA = 16807;     /* the multiplier */
static long     nM = 2147483647;/* the modulus == 2^31 - 1 */
static long     nQ = 127773;    /* the quotient nM / nA */
static long     nR = 2836;      /* the remainder nM % nA */

static double   dM = 2147483647.0;

/*
```

```
 * macros to control RNG and assure reproducible multi-stream
 * runs without the need for seed files. Keep track of invocations of
RNG
 * and always round-up to a known per-row boundary.
 */
/*
 * preferred solution, but not initializing correctly
 */
#define VSTR_MAX(len)           (long)(len / 5 + (len % 5 == 0)?0:1 +
1)
seed_t     Seed[MAX_STREAM + 1] =
{
   {PART,   1,         0,              1},
                        /* P_MFG_SD     0 */
   {PART,   46831694,   0, 1},
            /* P_BRND_SD    1 */
   {PART,   1841581359, 0, 1},
            /* P_TYPE_SD    2 */
   {PART,   1193163244, 0, 1},
            /* P_SIZE_SD    3 */
   {PART,   727633698,  0, 1},
            /* P_CNTR_SD    4 */
   {NONE,   933588178,  0, 1},
            /* P_RCST_SD    5  UNUSED 2-4-98 */
   {PART,   804159733,  0, RNG_PER_SENT * 3},     /*
P_CMNT_SD    6 */
   {PSUPP,  1671059989, 0, SUPP_PER_PART},     /* PS_QTY_SD
7 */
   {PSUPP,  1051288424, 0, SUPP_PER_PART},     /* PS_SCST_SD
8 */
   {PSUPP,  1961692154, 0, SUPP_PER_PART * RNG_PER_SENT *
20},     /* PS_CMNT_SD   9 */
   {ORDER,  1227283347, 0, 1},
               /* O_SUPP_SD    10 */
   {ORDER,  1171034773, 0, 1},
                        /* O_CLRK_SD    11 */
   {ORDER,  276090261,  0, RNG_PER_SENT * 8},  /* O_CMNT_SD
12 */
            {ORDER,  1066728069, 0, 1},
                        /* O_ODATE_SD   13 */
   {LINE,   209208115,  0, O_LCNT_MAX},       /* L_QTY_SD     14
*/
   {LINE,   554590007,  0, O_LCNT_MAX},       /* L_DCNT_SD    15
*/
   {LINE,   721958466,  0, O_LCNT_MAX},       /* L_TAX_SD     16
*/
   {LINE,   1371272478, 0, O_LCNT_MAX},       /* L_SHIP_SD    17
*/
   {LINE,   675466456,  0, O_LCNT_MAX},       /* L_SMODE_SD
18 */
   {LINE,   1808217256, 0, O_LCNT_MAX},     /* L_PKEY_SD    19
*/
   {LINE,   2095021727, 0, O_LCNT_MAX},     /* L_SKEY_SD    20
*/
   {LINE,   1769349045, 0, O_LCNT_MAX},     /* L_SDTE_SD    21
*/
   {LINE,   904914315,  0, O_LCNT_MAX},     /* L_CDTE_SD    22
*/
   {LINE,   373135028,  0, O_LCNT_MAX},     /* L_RDTE_SD    23
*/
   {LINE,   717419739,  0, O_LCNT_MAX},     /* L_RFLG_SD    24
*/
   {LINE,   1095462486, 0, O_LCNT_MAX * RNG_PER_SENT * 5},
/* L_CMNT_SD    25 */
   {CUST,   881155353,  0, 9},      /* C_ADDR_SD    26 */
   {CUST,   1489529863, 0, 1},      /* C_NTRG_SD    27 */
   {CUST,   1521138112, 0, 3},      /* C_PHNE_SD    28 */
   {CUST,   298370230,  0, 1},      /* C_ABAL_SD    29 */
   {CUST,   1140279430, 0, 1},      /* C_MSEG_SD    30 */
   {CUST,   1335826707, 0, RNG_PER_SENT * 12},     /*
C_CMNT_SD    31 */
   {SUPP,   706178559,  0, 9},      /* S_ADDR_SD    32 */
   {SUPP,   110356601,  0, 1},      /* S_NTRG_SD    33 */
   {SUPP,   884434366,  0, 3},      /* S_PHNE_SD    34 */
   {SUPP,   962338209,  0, 1},      /* S_ABAL_SD    35 */
   {SUPP,   1341315363, 0, RNG_PER_SENT * 11},     /*
S_CMNT_SD    36 */
   {PART,   709314158,  0, 92},     /* P_NAME_SD    37 */
   {ORDER,  591449447,  0, 1},      /* O_PRIO_SD    38 */
   {LINE,   431918286,  0, 1},     /* HVAR_SD      39 */
   {ORDER,  851767375,  0, 1},      /* O_CKEY_SD    40 */
   {NATION, 606179079,  0, RNG_PER_SENT * 16},     /*
N_CMNT_SD    41 */
   {REGION, 1500869201, 0, RNG_PER_SENT * 16},     /*
R_CMNT_SD    42 */
   {ORDER,  1434868289, 0, 1},      /* O_LCNT_SD    43 */
   {SUPP,   263032577,  0, 1},      /* BBB offset   44 */
   {SUPP,   753643799,  0, 1},      /* BBB type     45 */
   {SUPP,   202794285,  0, 1},      /* BBB comment  46 */
   {SUPP,   715851524,  0, 1}       /* BBB junk     47 */
};
```

shared.h

```
/*
 * $Header: shared.h 11-apr-2001.14:43:47 mpoess Exp $
 */

/* Copyright (c) Oracle Corporation 1999. All Rights Reserved. */

/* NOTE:  See 'header_template.doc' in the 'doc' dve under the 'forms'
     directory for the header file template that includes instructions.
*/

/*
  NAME
    <<<LOWERCASE_FILENAME>>>.h - <one-line expansion of the
name>

  DESCRIPTION
    <short description of component this file declares/defines>

  RELATED DOCUMENTS

  INSPECTION STATUS
    Inspection date:
    Inspection status:
    Estimated increasing cost defects per page:
    Rule sets:

  ACCEPTANCE REVIEW STATUS
    Review date:
    Review status:
    Reviewers:

  PUBLIC FUNCTION(S)
    <list of external functions declared/defined - with one-line
descriptions>

  PRIVATE FUNCTION(S)
    <list of static functions defined in .c file - with one-line
descriptions>

  EXAMPLES

  NOTES
    <other useful comments, qualifications, etc.>

  MODIFIED   (MM/DD/YY)
  mpoess      04/11/01 -
  mpoess      07/12/99 - exchange with new version July 8 1999
  mpoess      06/28/99 - Creation
  mpoess      06/28/99 - Creation

*/

/*
```

```
 *  Sccsid:     @(#)shared.h          2.1.8.1
 *
 */
#define N_CMNT_LEN      72
#define N_CMNT_MAX      152
#define R_CMNT_LEN      72
#define R_CMNT_MAX      152
#define  MONEY_SCL     0.01
#define  V_STR_HGH    1.6
#define  P_NAME_LEN    55
#define  P_MFG_LEN     25
#define  P_BRND_LEN    10
#define  P_TYPE_LEN    25
#define  P_CNTR_LEN    10
#define  P_CMNT_LEN    14
#define  P_CMNT_MAX    23
#define  S_NAME_LEN    25
#define  S_ADDR_LEN    25
#define  S_ADDR_MAX    40
#define  S_CMNT_LEN    63
#define  S_CMNT_MAX    101
#define  PS_CMNT_LEN  124
#define  PS_CMNT_MAX  199
#define  C_NAME_LEN    18
#define  C_ADDR_LEN    25
#define  C_ADDR_MAX    40
#define  C_MSEG_LEN    10
#define  C_CMNT_LEN    73
#define  C_CMNT_MAX    117
#define  O_OPRIO_LEN   15
#define  O_CLRK_LEN    15
#define  O_CMNT_LEN    49
#define  O_CMNT_MAX    79
#define  L_CMNT_LEN    27
#define  L_CMNT_MAX    44
#define  L_INST_LEN    25
#define  L_SMODE_LEN   10
#define  T_ALPHA_LEN   10
#define  DATE_LEN      13  /* long enough to hold either date format */
#define  NATION_LEN    25
#define  REGION_LEN    25
#define  PHONE_LEN     15
#define  MAXAGG_LEN    20    /* max component length for a agg str
*/
#define  P_CMNT_SD      6
#define  PS_CMNT_SD     9
#define  O_CMNT_SD     12
#define  C_ADDR_SD     26
#define  C_CMNT_SD     31
#define  S_ADDR_SD     32
#define  S_CMNT_SD     36
#define  L_CMNT_SD     25
```

speed_seed.c

```
#ifdef RCSID
static char *RCSid =
  "$Header: speed_seed.c 31-aug-99.12:22:59 mpoess Exp $ ";
#endif /* RCSID */

/* Copyright (c) Oracle Corporation 1999. All Rights Reserved. */

/*

  NAME
    speed_seed.c - <one-line expansion of the name>

  DESCRIPTION
    <short description of component this file declares/defines>

  PUBLIC FUNCTION(S)
    <list of external functions declared/defined - with one-line
descriptions>

  PRIVATE FUNCTION(S)
    <list of static functions defined in .c file - with one-line
descriptions>

  RETURNS
    <function return values, for .c file with single function>

  NOTES
    <other useful comments, qualifications, etc.>

  MODIFIED   (MM/DD/YY)
  mpoess      08/31/99 - add Jacks changes for version 1.1.0/1.2.0 (H/R)
as of
  mpoess      07/12/99 - exchange with new version July 8 1999
  mpoess      06/28/99 - Creation
  mpoess      06/28/99 - Creation

*/



/* @(#)speed_seed.c  2.1.8.2 */
#include <stdio.h>
#include <stdlib.h>
#include "dss.h"

/*  _tal long RandSeed = "Random^SeedFromTimestamp" (void); */

#define FAKE_V_STR(avg, sd, cnt) \
                ADVANCE_STREAM(sd, \
                        (long)(Seed[sd].boundary*cnt))
#define ADVANCE_STREAM(stream_id, num_calls) \
        NthElement(num_calls, &Seed[stream_id].value)

#define MAX_COLOR 92
long name_bits[MAX_COLOR / BITS_PER_LONG];
extern seed_t Seed[];

/* WARNING!  This routine assumes the existence of 64-bit
*/
/* integers.  The notation used here- "HUGE" is *not* ANSI standard.
*/
/* Hopefully, you have this extension as well.  If not, use whatever      */
/* nonstandard trick you need to in order to get 64 bit integers.         */
/* The book says that this will work if MAXINT for the type you choose
*/
/* is at least 2**46  - 1, so 64 bits is more than you *really* need      */

static DSS_HUGE Multiplier = 16807;      /* or whatever nonstandard
*/
static DSS_HUGE Modulus =  2147483647;   /* trick you use to get 64
bit int */

/* Advances value of Seed after N applications of the random number
generator
  with multiplier Mult and given Modulus.
  NthElement(Seed[],count);

  Theory:  We are using a generator of the form
      X_n = [Mult * X_(n-1)]  mod Modulus.    It turns out that
      X_n = [(Mult ** n) X_0] mod Modulus.
  This can be computed using a divide-and-conquer technique, see
  the code below.

  In words, this means that if you want the value of the Seed after n
  applications of the generator,  you multiply the initial value of the
  Seed by the "super multiplier" which is the basic multiplier raised
  to the nth power, and then take mod Modulus.
*/
```

```
/* Nth Element of sequence starting with StartSeed */
/* Warning, needs 64-bit integers */
#ifdef SUPPORT_64BITS
void NthElement (long N, long *StartSeed)
  {
  DSS_HUGE Z;
  DSS_HUGE Mult;
  static int ln=-1;
  int i;

  if ((verbose > 0) && ++ln % 1000 == 0)
     {
     i = ln % LN_CNT;
     fprintf(stderr, "%c\b", lnoise[i]);
     }
  Mult = Multiplier;
  Z = (DSS_HUGE) *StartSeed;
  while (N > 0 )
    {
    if (N % 2 != 0)    /* testing for oddness, this seems portable */
      Z = (Mult * Z) % Modulus;
    N = N / 2;         /* integer division, truncates */
    Mult = (Mult * Mult) % Modulus;
    }
  *StartSeed = (long)Z;

  return;
  }
#else
/* add 32 bit version of NthElement HERE */
/*
 *    MODMULT.C
 *    R. M. Shelton -- Unisys
 *    July 26, 1995
 *
 *    RND_seed:  Computes the nth seed in the total sequence
 *    RND_shift:  Shifts a random number by a given number of seeds
 *    RND_ModMult:  Multiplies two numbers mod (2^31 - 1)
 *
 */



#include <math.h>
#include <stdio.h>       /* required only for F_FatalError */

typedef signed long RND;
typedef unsigned long URND;

#define FatalError(e)  F_FatalError( (e), __FILE__, __LINE__ )
void F_FatalError( int x, char *y, int z ) {fprintf(stderr, "Bang!\n");}


/* Prototypes */
RND RND_seed( RND );
RND RND_shift( RND, RND );
static RND RND_ModMult( RND, RND );



RND
RND_seed ( RND Order            )
{
static const RND TopMask = 0x40000000;
RND Mask;
RND Result;


if (Order <= -Modulus || Order >= Modulus)
  FatalError(1023);

if (Order < 0) Order = Modulus - 1L + Order;

Mask = TopMask;
Result = 1L;

while (Mask > Order) Mask >>= 1;

while (Mask > 0)
  {
  if (Mask & Order)
    {
    Result = RND_ModMult( Result, Result);
    Result = RND_ModMult( Result, Multiplier );
    }
  else
    {
    Result = RND_ModMult( Result, Result );
    }
  Mask >>= 1;
  }

return (Result);

}  /*  RND_seed  */



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

   RND_shift:  Shifts a random number by a given number of seeds

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

void
NthElement ( long Shift, long *Seed)

{
  RND Power;
  static int ln=-1;
  int i;

  if ((verbose > 0) && ++ln % 100 == 0)
     {
     i = (ln/100) % LN_CNT;
     fprintf(stderr, "%c\b", lnoise[i]);
     }


if (*Seed <= 0 || *Seed >= Modulus)
  FatalError(1023);
if (Shift <= -Modulus || Shift >= Modulus)
  FatalError(1023);

Power = RND_seed( Shift );

*Seed = RND_ModMult( *Seed, Power );

return;
}  /*  RND_shift  */



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

   RND_ModMult:  Multiplies two numbers mod (2^31 - 1)

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

static RND
RND_ModMult ( RND nA, RND nB)
```

```
{

static const double dTwoPowPlus31 = 2147483648.;
static const double dTwoPowMinus31 = 1./2147483648.;
static const double dTwoPowPlus15 = 32768.;
static const double dTwoPowMinus15 = 1./32768.;
static const RND    nLowMask = 0xFFFFL;
static const URND   ulBit31 = 1uL << 31;

double dAH, dAL, dX, dY, dZ, dW;
RND    nH, nL;
URND   ulP, ulQ, ulResult;

nL = nB & nLowMask;
nH = (nB - nL) >> 16;
dAH = (double)nA * (double)nH;
dAL = (double)nA * (double)nL;
dX = floor( dAH * dTwoPowMinus15 );
dY = dAH - dX*dTwoPowPlus15;
dZ = floor( dAL * dTwoPowMinus31 );
dW = dAL - dZ*dTwoPowPlus31;

ulQ = (URND)dW + ((URND)dY << 16);
ulP = (URND)dX + (URND)dZ;
if (ulQ & ulBit31) { ulQ -= ulBit31; ulP++; }

ulResult = ulP + ulQ;
if (ulResult & ulBit31) { ulResult -= ulBit31; ulResult++; }

return (RND)ulResult;
}
#endif /* SUPPORT_64BITS */

/* updates Seed[column] using the a_rnd algorithm */
void
fake_a_rnd(int min, int max, int column)
{
  long len, itcount;
  RANDOM(len, (long)min, (long)max, (long)column);
  if (len % 5L == 0)
    itcount = len/5;
  else itcount = len/5 + 1L;
  NthElement(itcount, &Seed[column].usage);
  return;
}


long
sd_part(int child, long skip_count)
{
  int i;

  for (i=P_MFG_SD; i<= P_CNTR_SD; i++)
     ADVANCE_STREAM(i, skip_count);

  FAKE_V_STR(P_CMNT_LEN, P_CMNT_SD, skip_count);
  ADVANCE_STREAM(P_NAME_SD, skip_count * 92);

  return(0L);
}

long
sd_line(int child, long skip_count)
        {
        int i,j;

        for (j=0; j < O_LCNT_MAX; j++)
                {
                for (i=L_QTY_SD; i<= L_RFLG_SD; i++)
                        ADVANCE_STREAM(i, skip_count);
                }

        FAKE_V_STR(L_CMNT_LEN, L_CMNT_SD, skip_count);
        /* need to special case this as the link between master and
detail */
        if (child == 1)
                {
                ADVANCE_STREAM(O_ODATE_SD,
skip_count);
                ADVANCE_STREAM(O_LCNT_SD,
skip_count);
                }

        return(0L);
        }

long
sd_order(int child, long skip_count)
{
  ADVANCE_STREAM(O_LCNT_SD, skip_count);
  ADVANCE_STREAM(O_CKEY_SD, skip_count);
  FAKE_V_STR(O_CMNT_LEN, O_CMNT_SD, skip_count);
  ADVANCE_STREAM(O_SUPP_SD, skip_count);
  ADVANCE_STREAM(O_CLRK_SD, skip_count);
  ADVANCE_STREAM(O_PRIO_SD, skip_count);
  ADVANCE_STREAM(O_ODATE_SD, skip_count);

  return (0L);
}

long
sd_psupp(int child, long skip_count)
        {
        int j;

        for (j=0; j < SUPP_PER_PART; j++)
                {
                ADVANCE_STREAM(PS_QTY_SD,
skip_count);
                ADVANCE_STREAM(PS_SCST_SD,
skip_count);
                }
        FAKE_V_STR(PS_CMNT_LEN, PS_CMNT_SD,
skip_count);

        return(0L);
        }

long
sd_cust(int child, long skip_count)
{

  FAKE_V_STR(C_ADDR_LEN, C_ADDR_SD, skip_count);
  FAKE_V_STR(C_CMNT_LEN, C_CMNT_SD, skip_count);
  ADVANCE_STREAM(C_NTRG_SD, skip_count);
  ADVANCE_STREAM(C_PHNE_SD, 3L * skip_count);
  ADVANCE_STREAM(C_ABAL_SD, skip_count);
  ADVANCE_STREAM(C_MSEG_SD, skip_count);
  return(0L);
}

long
sd_supp(int child, long skip_count)
{
  ADVANCE_STREAM(S_NTRG_SD, skip_count);
  ADVANCE_STREAM(S_PHNE_SD, 3L * skip_count);
  ADVANCE_STREAM(S_ABAL_SD, skip_count);
  FAKE_V_STR(S_ADDR_LEN, S_ADDR_SD, skip_count);
  FAKE_V_STR(S_CMNT_LEN, S_CMNT_SD, skip_count);
  ADVANCE_STREAM(BBB_CMNT_SD, skip_count);
  ADVANCE_STREAM(BBB_JNK_SD, skip_count);
```

```
  ADVANCE_STREAM(BBB_OFFSET_SD, skip_count);
  ADVANCE_STREAM(BBB_TYPE_SD, skip_count);      /* avoid
one trudge */

  return(0L);
}
```

text.c

```
#ifdef RCSID
static char *RCSid =
  "$Header: text.c 11-apr-2001.14:42:01 mpoess Exp $ ";
#endif /* RCSID */

/* Copyright (c) Oracle Corporation 1999. All Rights Reserved. */

/*

  NAME
    text.c version: 2.1.6.1

  DESCRIPTION
    pseaudo text generator for use in DBGEN 2.0

  NOTES


  MODIFIED   (MM/DD/YY)
  mpoess     04/11/01 -
  mpoess     07/12/99 - exchange with new version July 8 1999
  mpoess     06/28/99 - Creation
  mpoess     06/28/99 - Creation

/* @(#)text.c              2.1.8.1 */
/*
 * text.c --- pseaudo text generator for use in DBGEN 2.0
 *
 * Defined Routines:
 *                      dbg_text() -- select and translate a sentance form
 */

#ifdef TEST
#define DECLARER
#endif /* TEST */

#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>
#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 NOSYSMETRICS
#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 "dsstypes.h"

/*
 * txt_vp() --
 *                      generate a verb phrase by
 *                      1) selecting a verb phrase form
 *                      2) parsing it to select parts of speech
 *                      3) selecting appropriate words
 *                      4) adding punctuation as required
 *
 *          Returns: length of generated phrase
 *          Called By: txt_sentence()
 *          Calls: pick_str()
 */
static int
txt_vp(char *dest, int sd)
{
        char syntax[MAX_GRAMMAR_LEN + 1],
                *cptr,
                *parse_target;
        distribution *src;
        int i,
                res = 0;


        pick_str(&vp, sd, &syntax[0]);
        parse_target = syntax;
        while ((cptr = strtok(parse_target, " ")) != NULL)
        {
                src = NULL;
                switch(*cptr)
                {
                case 'D':
                        src = &adverbs;
                        break;
                case 'V':
                        src = &verbs;
                        break;
                case 'X':
                        src = &auxillaries;
                        break;
                }       /* end of POS switch statement */
                i = pick_str(src, sd, dest);
                i = strlen(DIST_MEMBER(src, i));
                dest += i;
                res += i;
                if (*(++cptr))          /* miscelaneous fillagree,
like punctuation */
                {
                        dest += 1;
                        res += 1;
                        *dest = *cptr;
                }
                *dest = ' ';
                dest++;
                res++;
                parse_target = NULL;
        }       /* end of while loop */
```

```
	return(res);
}

/*
 * txt_np() --
 *			generate a noun phrase by
 *			1) selecting a noun phrase form
 *			2) parsing it to select parts of speech
 *			3) selecting appropriate words
 *			4) adding punctuation as required
 *
 *		Returns: length of generated phrase
 *		Called By: txt_sentence()
 *		Calls: pick_str(),
 */
static int
txt_np(char *dest, int sd)
{
	char syntax[MAX_GRAMMAR_LEN + 1],
			*cptr,
			*parse_target;
	distribution *src;
	int i,
			res = 0;


	pick_str(&np, sd, &syntax[0]);
	parse_target = syntax;
	while ((cptr = strtok(parse_target, " ")) != NULL)
	{
			src = NULL;
			switch(*cptr)
			{
			case 'A':
					src = &articles;
					break;
			case 'J':
					src = &adjectives;
					break;
			case 'D':
					src = &adverbs;
					break;
			case 'N':
					src = &nouns;
					break;
			}		/* end of POS switch statement */
			i = pick_str(src, sd, dest);
			i = strlen(DIST_MEMBER(src, i));
			dest += i;
			res += i;
			if (*(++cptr))		/* miscelaneous fillagree,
like punctuation */
			{
					*dest = *cptr;
					dest += 1;
					res += 1;
			}
			*dest = ' ';
			dest++;
			res++;
			parse_target = NULL;
	}		/* end of while loop */

	return(res);
}

/*
 * txt_sentence() --
 *			generate a sentence by
 *			1) selecting a sentence form
 *			2) parsing it to select parts of speech or phrase
types
 *			3) selecting appropriate words
 *			4) adding punctuation as required
 *
 *		Returns: length of generated sentence
 *		Called By: dbg_text()
 *		Calls: pick_str(), txt_np(), txt_vp()
 */
static int
txt_sentence(char *dest, int sd)
{
	char syntax[MAX_GRAMMAR_LEN + 1],
			*cptr;
	int i,
			res = 0,
			len = 0;


	pick_str(&grammar, sd, syntax);
	cptr = syntax;

next_token:		/* I hate goto's, but can't seem to have parent and
child use strtok() */
	while (*cptr && *cptr == ' ')
			cptr++;
	if (*cptr == '\0')
			goto done;
	switch(*cptr)
			{
			case 'V':
					len = txt_vp(dest, sd);
					break;
			case 'N':
					len = txt_np(dest, sd);
					break;
			case 'P':
					i = pick_str(&prepositions, sd, dest);
					len =
strlen(DIST_MEMBER(&prepositions, i));
					strcpy((dest + len), " the ");
					len += 5;
					len += txt_np(dest + len, sd);
					break;
			case 'T':
					i = pick_str(&terminators, sd, --dest);
/*terminators should abut previous word */
					len =
strlen(DIST_MEMBER(&terminators, i));
					break;
			}		/* end of POS switch statement */
			dest += len;
			res += len;
			cptr++;
			if (*cptr && *cptr != ' ')		/* miscelaneous
fillagree, like punctuation */
			{
					dest += 1;
					res += 1;
					*dest = *cptr;
			}
			goto next_token;
done:
	*dest = '\0';
	return(--res);
}

/*
 * dbg_text() --
 *			produce ELIZA-like text of random, bounded
length, truncating the last
```

```
 *                      generated sentence as required
 */
int
dbg_text(char *tgt, int min, int max, int sd)
{
        long length = 0;
        int wordlen = 0,
                needed,
                s_len;
        char sentence[MAX_SENT_LEN + 1];

        RANDOM(length, min, max, sd);

        while (wordlen < length)
        {
                s_len = txt_sentence(sentence, sd);
                if ( s_len < 0)
                        INTERNAL_ERROR("Bad sentence
formation");
                needed = length - wordlen;
                if (needed >= s_len + 1)        /* need the
entire sentence */
                {
                        strcpy(tgt, sentence);
                        tgt += s_len;
                        wordlen += s_len + 1;
                        *(tgt++) = ' ';
                }
                else /* chop the new sentence off to match the
length target */
                {
                        sentence[needed] = '\0';
                        strcpy(tgt, sentence);
                        wordlen += needed;
                        tgt += needed;
                }
        }
        *tgt = '\0';

        return(wordlen);
}

#ifdef TEST
tdef tdefs = { NULL };

main()
{
        char prattle[401];

        read_dist (env_config (DIST_TAG, DIST_DFLT), "nouns",
&nouns);
        read_dist (env_config (DIST_TAG, DIST_DFLT), "verbs",
&verbs);
        read_dist (env_config (DIST_TAG, DIST_DFLT),
"adjectives", &adjectives);
        read_dist (env_config (DIST_TAG, DIST_DFLT), "adverbs",
&adverbs);
        read_dist (env_config (DIST_TAG, DIST_DFLT),
"auxillaries", &auxillaries);
        read_dist (env_config (DIST_TAG, DIST_DFLT),
"terminators", &terminators);
        read_dist (env_config (DIST_TAG, DIST_DFLT), "articles",
&articles);
        read_dist (env_config (DIST_TAG, DIST_DFLT),
"prepositions", &prepositions);
        read_dist (env_config (DIST_TAG, DIST_DFLT),
"grammar", &grammar);
        read_dist (env_config (DIST_TAG, DIST_DFLT), "np",
&np);
        read_dist (env_config (DIST_TAG, DIST_DFLT), "vp",
&vp);

        while (1)
        {
                dbg_text(&prattle[0], 300, 400, 0);
                printf("<%s>\n", prattle);
        }

        return(0);
}
#endif /* TEST */
```

tpcd.h

```
/*
 * $Header: tpcd.h 11-apr-2001.14:40:53 mpoess Exp $
 */

/* Copyright (c) Oracle Corporation 1999. All Rights Reserved. */

/* NOTE:  See 'header_template.doc' in the 'doc' dve under the 'forms'
     directory for the header file template that includes instructions.
*/

/*
   NAME
     <<<LOWERCASE_FILENAME>>>.h - <one-line expansion of the
name>

   DESCRIPTION
     <short description of component this file declares/defines>

   RELATED DOCUMENTS

   INSPECTION STATUS
     Inspection date:
     Inspection status:
     Estimated increasing cost defects per page:
     Rule sets:

   ACCEPTANCE REVIEW STATUS
     Review date:
     Review status:
     Reviewers:

   PUBLIC FUNCTION(S)
     <list of external functions declared/defined - with one-line
descriptions>

   PRIVATE FUNCTION(S)
     <list of static functions defined in .c file - with one-line
descriptions>

   EXAMPLES

   NOTES
     <other useful comments, qualifications, etc.>

   MODIFIED   (MM/DD/YY)
   mpoess      04/11/01 -
   mpoess      07/12/99 - exchange with new version July 8 1999
   mpoess      07/08/99 - add rownum <= support
   mpoess      06/28/99 - add defines for Oracle
   mpoess      06/28/99 - Creation
   mpoess      06/28/99 - Creation

*/
/************************************************************
*******
 *  Title: tpcd.h for TPC D
 *  Sccsid: @(#)tpcd.h              2.1.8.1
 *  Description:
 *  X
 *

************************************************************
******
```

```
 */

#define ORACLE 1

/*************************************************************
*******
 *  Title: tpcd.h for TPC D
 *  Sccsid: @(#)tpcd.h              2.1.6.1
 *  Description:
 *  X
 *

*************************************************************
******
 */
#define DFLT            0x0001
#define OUTPUT          0x0002
#define EXPLAIN         0x0004
#define DBASE           0x0008
#define VERBOSE         0x0010
#define TIMING          0x0020
#define LOG             0x0040
#define QUERY           0x0080
#define REFRESH         0x0100
#define ANSI            0x0200
#define SEED            0x0400
#define COMMENT         0x0800
#define INIT            0x1000
#define TERMINATE       0x2000
#define DFLT_NUM        0x4000

/*
 * general defines
 */
#define VTAG            ':'         /* flags a variable substitution */
#define ofp             stdout      /* make the routine a filter */
#define QDIR_TAG        "DSS_QUERY"  /* variable to point to queries
*/
#define QDIR_DFLT       "."         /* and its default */

/*
 * database portability defines
 */
#ifdef DB2
#define GEN_QUERY_PLAN  "SET CURRENT EXPLAIN
SNAPSHOT ON;"
#define START_TRAN      ""
#define END_TRAN        "COMMIT WORK;"
#define SET_OUTPUT      ""
#define SET_ROWCOUNT    "--#SET ROWS_FETCH %d\n"
#define SET_DBASE       "CONNECT TO %s ;\n"
#endif

#ifdef INFORMIX
#define GEN_QUERY_PLAN  "SET EXPLAIN ON;"
#define START_TRAN      "BEGIN WORK;"
#define END_TRAN        "COMMIT WORK;"
#define SET_OUTPUT      "OUTPUT TO "
#define SET_ROWCOUNT    "FIRST %d"
#define SET_DBASE       "database %s ;\n"
#endif

#ifdef        SQLSERVER
#define GEN_QUERY_PLAN  "set showplan on\nset noexec on\ngo\n"
#define START_TRAN      "begin transaction\ngo\n"
#define END_TRAN        "commit transaction\ngo\n"
#define SET_OUTPUT      ""
#define SET_ROWCOUNT    "set rowcount %d\ngo\n\n"
#define SET_DBASE       "use %s\ngo\n"
#endif

#ifdef        SYBASE
#define GEN_QUERY_PLAN  "set showplan on\nset noexec on\ngo\n"
#define START_TRAN      "begin transaction\ngo\n"
#define END_TRAN        "commit transaction\ngo\n"
#define SET_OUTPUT      ""
#define SET_ROWCOUNT    "set rowcount %d\ngo\n\n"
#define SET_DBASE       "use %s\ngo\n"
#endif

#ifdef TDAT
#define GEN_QUERY_PLAN  "EXPLAIN"
#define START_TRAN      "BEGIN TRANSACTION"
#define END_TRAN        "END TRANSACTION"
#define SET_OUTPUT      ".SET FORMAT OFF\n.EXPORT REPORT
file="
#define SET_ROWCOUNT    ".SET RETCANCEL ON\n.SET
RETLIMIT %d\n"
#define SET_DBASE       ".LOGON %s\n"
#endif

#ifdef ORACLE
#define GEN_QUERY_PLAN  ""
#define START_TRAN      ""
#define END_TRAN        ""
#define SET_OUTPUT      ""
#define SET_ROWCOUNT    "where rownum <= %d;\n"
#define SET_DBASE       ""
#endif


#define MAX_VARS      8 /* max number of host vars in any query */
#define QLEN_MAX   2048 /* max length of any query */
#define QUERIES_PER_SET 22
#define MAX_PIDS 50

EXTERN int flags;
EXTERN int s_cnt;
EXTERN char *osuff;
EXTERN int stream;
EXTERN char *lfile;
EXTERN char *ifile;
EXTERN char *tfile;

#define MAX_PERMUTE     41
#ifdef DECLARER
int rowcnt_dflt[QUERIES_PER_SET + 1] =
    {-1,-1,100,10,-1,-1,-1,-1,-1,-1,20,-1,-1,-1,-1,-1,-1,-1,100,-1,-1,100,-
1};
int rowcnt;
#define SEQUENCE(stream, query) permutation[stream %
MAX_PERMUTE][query - 1]
#else
extern int rowcnt_dflt[];
extern int rowcnt;
#endif
```

varsub.c

```
#ifdef RCSID
static char *RCSid =
   "$Header: varsub.c 11-apr-2001.14:37:26 mpoess Exp $ ";
#endif /* RCSID */

/* Copyright (c) Oracle Corporation 1999. All Rights Reserved. */

/*

   NAME
     varsub.c - <one-line expansion of the name>

   DESCRIPTION
```

```
    <short description of component this file declares/defines>

  PUBLIC FUNCTION(S)
    <list of external functions declared/defined - with one-line
descriptions>

  PRIVATE FUNCTION(S)
    <list of static functions defined in .c file - with one-line
descriptions>

  RETURNS
    <function return values, for .c file with single function>

  NOTES
    <other useful comments, qualifications, etc.>

  MODIFIED   (MM/DD/YY)
  mpoess     04/11/01 -
  mpoess     08/31/99 - add Jacks changes for version 1.1.0/1.2.0 (H/R)
as of
  mpoess     07/12/99 - exchange with new version July 8 1999
  mpoess     06/28/99 - Creation
  mpoess     06/28/99 - Creation

*/


/* Sccsid:     @(#)varsub.c          2.1.8.3 */
#include <stdio.h>
#ifndef _POSIX_SOURCE
#include <malloc.h>
#endif /* POSIX_SOURCE */
#if (defined(_POSIX_)||!defined(WIN32))
#include <unistd.h>
#endif /* WIN32 */
#include <string.h>
#include "config.h"
#include "dss.h"
#include "tpcd.h"
#ifdef ADHOC
#include "adhoc.h"
extern adhoc_t adhocs[];
#endif /* ADHOC */

#define MAX_PARAM               10                      /* maximum
number of parameter substitutions in a query */

extern long Seed[];
extern char **asc_date;
extern double flt_scale;
extern distribution q13a, q13b;
long *permute(long *set, int cnt, long stream);

long brands[25] = {11,12,13,14,15,21,22,23,24,25,31,32,33,34,35,
                                        41,42,43,44,45,51,52,53,54,55};
long sizes[50] = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,

            21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,4
0,

            41,42,43,44,45,46,47,48,49,50};
long ccode[25] =
{10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,3
2,33,34};
char *defaults[24][11] =
{
   {"90",           NULL,               NULL,
      NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL},  /*
1 */
   {"15",           "BRASS",            "EUROPE",
            NULL, NULL, NULL, NULL, NULL, NULL, NULL,
NULL},  /* 2 */
   {"BUILDING",     "1995-03-15",       NULL,
            NULL, NULL, NULL, NULL, NULL, NULL, NULL,
NULL},  /* 3 */
   {"1993-07-01",     NULL,               NULL,
            NULL, NULL, NULL, NULL, NULL, NULL, NULL,
NULL},  /* 4 */
   {"ASIA",         "1994-01-01",       NULL,
            NULL, NULL, NULL, NULL, NULL, NULL, NULL,
NULL},  /* 5 */
   {"1994-01-01",     ".06",              "24",
            NULL, NULL, NULL, NULL, NULL, NULL, NULL,
NULL},  /* 6 */
   {"FRANCE",       "GERMANY",          NULL,
            NULL, NULL, NULL, NULL, NULL, NULL, NULL,
NULL},  /* 7 */
   {"BRAZIL",       "AMERICA",     "ECONOMY ANODIZED
STEEL",
            NULL, NULL, NULL, NULL, NULL, NULL, NULL,
NULL},/* 8 */
   {"green",        NULL,               NULL,
            NULL, NULL, NULL, NULL, NULL, NULL, NULL,
NULL},  /* 9 */
   {"1993-10-01",     NULL,               NULL,
            NULL, NULL, NULL, NULL, NULL, NULL, NULL,
NULL},  /* 10 */
   {"GERMANY",      "0.0001",           NULL,
            NULL, NULL, NULL, NULL, NULL, NULL, NULL,
NULL},  /* 11 */
   {"MAIL",         "SHIP",             "1994-01-01",
            NULL, NULL, NULL, NULL, NULL, NULL, NULL,
NULL},  /* 12 */
   {"special", "requests",              NULL,
            NULL, NULL, NULL, NULL, NULL, NULL, NULL,
NULL},  /* 13 */
   {"1995-09-01",     NULL,               NULL,
            NULL, NULL, NULL, NULL, NULL, NULL, NULL,
NULL},  /* 14 */
   {"1996-01-01",     NULL,               NULL,
            NULL, NULL, NULL, NULL, NULL, NULL, NULL,
NULL},  /* 15 */
   {"Brand#45",     "MEDIUM POLISHED", "49",
            "14","23","45","19","3","36","9", NULL}, /* 16 */
   {"Brand#23",     "MED BOX",          NULL,
            NULL, NULL, NULL, NULL, NULL, NULL, NULL,
NULL},  /* 17 */
   {"300", NULL, NULL, NULL, NULL, NULL, NULL, NULL, NULL,
NULL, NULL},  /* 18 */
   {"Brand#12", "Brand#23", "Brand#34", "1", "10", "20", NULL,
NULL, NULL, NULL, NULL, NULL}, /* 19 */
   {"forest", "1994-01-01", "CANADA", NULL, NULL, NULL, NULL,
NULL, NULL, NULL, NULL, NULL},  /* 20 */
   {"SAUDI ARABIA", NULL, NULL, NULL, NULL, NULL, NULL,
NULL, NULL, NULL, NULL, NULL},  /* 21 */
   {"13","31","23", "29", "30", "18", "17", NULL, NULL, NULL,
NULL},  /* 22 */
   {NULL,NULL,NULL,NULL, NULL, NULL, NULL, NULL, NULL, NULL,
NULL, NULL},  /* UF1 */
   {NULL,NULL,NULL,NULL, NULL, NULL, NULL, NULL, NULL, NULL,
NULL, NULL},  /* UF2 */
};
void
varsub(int qnum, int vnum, int flags)
{
   static char param[11][128];
   static FILE *lfp = NULL;
            long *lptr;
   char *ptr;
   int i = 0,
      tmp_date;
            long tmp1,
                        tmp2;

   if (vnum == 0)
   {
```

```
                if ((flags & DFLT) == 0)
                {
                        switch(qnum)
                        {
                        case 1:
                                sprintf(param[1], "%d",
UnifInt((long)60,(long)120,(long)qnum));
                                param[2][0] = '\0';
                                break;
                        case 2:
                                sprintf(param[1], "%d",

        UnifInt((long)P_SIZE_MIN, (long)P_SIZE_MAX, qnum));
                                pick_str(&p_types_set,
qnum, param[3]);
                                ptr = param[3] +
strlen(param[3]);
                                while (*(ptr - 1) != ' ') ptr--;
                                strcpy(param[2], ptr);
                                pick_str(&regions, qnum,
param[3]);
                                param[4][0] = '\0';
                                break;
                        case 3:
                                pick_str(&c_mseg_set,
qnum, param[1]);
                                /*
                                * pick a random offset
within the month of march and add the
                                * appropriate magic
numbers to position the output functions
                                * at the start of March '95
                                */
                                tmp_date =
UnifInt((long)0, (long)30, (long)qnum);
                                strcpy(param[2],
*(asc_date + tmp_date + 1155));
                                param[3][0] = '\0';
                                break;
                        case 4:
                                tmp_date =
UnifInt(1,58,qnum);
                                sprintf(param[1],"19%02d-
%02d-01",
                                                93 +
tmp_date/12, tmp_date%12 + 1);
                                param[2][0] = '\0';
                                break;
                        case 5:
                                pick_str(&regions, qnum,
param[1]);
                                tmp_date =
UnifInt((long)93,(long)97,(long)qnum);
                                sprintf(param[2], "19%d-
01-01", tmp_date);
                                param[3][0] = '\0';
                                break;
                        case 6:
                                tmp_date =
UnifInt(93,97,qnum);
        sprintf(param[1], "19%d-01-01", tmp_date);
                                sprintf(param[2], "0.0%d",
UnifInt(2, 9, qnum));
                                sprintf(param[3], "%d",
UnifInt((long)24, (long)25, (long)qnum));
                                param[4][0] = '\0';
                                break;
                        case 7:
                                tmp_date =
pick_str(&nations2, qnum, param[1]);
                                while (pick_str(&nations2,
qnum, param[2]) == tmp_date);
                                param[3][0] = '\0';
                                break;
                        case 8:
                                tmp_date =
pick_str(&nations2, qnum, param[1]);
                                tmp_date =
nations.list[tmp_date].weight;
                                strcpy(param[2],
regions.list[tmp_date].text);
                                pick_str(&p_types_set,
qnum, param[3]);
                                param[4][0] = '\0';
                                break;
                        case 9:
                                pick_str(&colors, qnum,
param[1]);
                                param[2][0] = '\0';
                                break;
                        case 10:
                                tmp_date =
UnifInt(1,24,qnum);
                                sprintf(param[1],"19%02d-
%02d-01",
                                                93 +
tmp_date/12, tmp_date%12 + 1);
                                param[2][0] = '\0';
                                break;
                        case 11:
                                pick_str(&nations2, qnum,
param[1]);
                                sprintf(param[2],
"%11.10f", Q11_FRACTION / flt_scale );
                                param[3][0] = '\0';
                                break;
                        case 12:
                                tmp_date =
pick_str(&l_smode_set, qnum, param[1]);
                                while (tmp_date ==
pick_str(&l_smode_set, qnum, param[2]));
                                tmp_date =
UnifInt(93,97,qnum);
                                sprintf(param[3], "19%d-
01-01", tmp_date);
                                param[4][0] = '\0';
                                break;
                        case 13:
                                pick_str(&q13a, qnum,
param[1]);
                                pick_str(&q13b, qnum,
param[2]);
                                param[3][0] = '\0';
                                break;
                        case 14:
                                tmp_date =
UnifInt(1,60,qnum);
                                sprintf(param[1],"19%02d-
%02d-01",
                                                93 +
tmp_date/12, tmp_date%12 + 1);
                                param[2][0] = '\0';
                                break;
                        case 15:
                                tmp_date =
UnifInt(1,58,qnum);
                                sprintf(param[1],"19%02d-
%02d-01",
                                                93 +
tmp_date/12, tmp_date%12 + 1);
                                param[2][0] = '\0';
```

```
                break;
        case 16:
                tmp1 = UnifInt(1, 5,
qnum);
                tmp2 = UnifInt(1, 5,
qnum);
                sprintf(param[1],
"Brand#%d%d", tmp1, tmp2);
                pick_str(&p_types_set,
qnum, param[2]);
                ptr = param[2] +
strlen(param[2]);
                while (*(--ptr) != ' ');
                *ptr = '\0';
                lptr = &sizes[0];
                for (i=3; i <=
MAX_PARAM; i++)
                {
                        sprintf(param[i],
"%ld", *permute(lptr,50,qnum)+1);
                        lptr = (long
*)NULL;
                }
                break;
        case 17:
                tmp1 = UnifInt(1, 5,
qnum);
                tmp2 = UnifInt(1, 5,
qnum);
                sprintf(param[1],
"Brand#%d%d", tmp1, tmp2);
                pick_str(&p_cntr_set,
qnum, param[2]);
                param[3][0] = '\0';
                break;
        case 18:
                sprintf(param[1], "%ld",
UnifInt(312, 315, qnum));
                param[2][0] = '\0';
                break;
        case 19:
                tmp1 = UnifInt(1, 5,
qnum);
                tmp2 = UnifInt(1, 5,
qnum);
                sprintf(param[1],
"Brand#%d%d", tmp1, tmp2);
                tmp1 = UnifInt(1, 5,
qnum);
                tmp2 = UnifInt(1, 5,
qnum);
                sprintf(param[2],
"Brand#%d%d", tmp1, tmp2);
                tmp1 = UnifInt(1, 5,
qnum);
                tmp2 = UnifInt(1, 5,
qnum);
                sprintf(param[3],
"Brand#%d%d", tmp1, tmp2);
                sprintf(param[4], "%ld",
UnifInt(1, 10, qnum));
                sprintf(param[5], "%ld",
UnifInt(10, 20, qnum));
                sprintf(param[6], "%ld",
UnifInt(20, 30, qnum));
                param[7][0] = '\0';
                break;
        case 20:
                pick_str(&colors, qnum,
param[1]);
                tmp_date =
UnifInt(93,97,qnum);
                sprintf(param[2], "19%d-
01-01", tmp_date);
                pick_str(&nations2, qnum,
param[3]);
                param[4][0] = '\0';
                break;
        case 21:
                pick_str(&nations2, qnum,
param[1]);
                param[2][0] = '\0';
                break;
        case 22:
                lptr = &ccode[0];
                for (i=0; i <= 7; i++)
                {
                        sprintf(param[i+1], "%ld", 10 + *permute(lptr,25, qnum));
                        lptr = (long
*)NULL;
                }
                param[8][0] = '\0';
                break;
        case 23:
        case 24:
                break;
        default:
                fprintf(stderr,
                        "No variable
definitions available for query %d\n",
                        qnum);
                return;
        }
    }

    if (flags & LOG)
            {
        if (lfp == NULL)
                    {
            lfp = fopen(lfile, "a");
            OPEN_CHECK(lfp, lfile);
                    }
        fprintf(lfp, "%d", qnum);
        for (i=1; i <= 10; i++)
            if (flags & DFLT)
                {
                    if (defaults[qnum - 1][i - 1]
== NULL)
                        break;
                    else
                        fprintf(lfp,
"\t%s", defaults[qnum - 1][i - 1]);
                }
            else
                {
                    if (param[i][0] == '\0')
                        break;
                    else
                        fprintf(lfp,
"\t%s", param[i]);
                }
            fprintf(lfp, "\n");
          }
    }
    else
            {
        if (flags & DFLT)
                    {
            /* to allow -d to work at all scale factors */
            if (qnum == 11 && vnum == 2)
                fprintf(ofp, "%11.10f", Q11_FRACTION/flt_scale);
            else
                if (defaults[qnum - 1][vnum - 1])
                    fprintf(ofp, "%s", defaults[qnum - 1][vnum - 1]);
```

```
            else
                                                    fprintf(stderr,
                                                    "Bad default
request (q: %d, p: %d)\n",
                                                    qnum, vnum);
                        }
        else
                        {
          if (param[vnum] && vnum <= MAX_PARAM)
             fprintf(ofp, "%s", param[vnum]);
          else
                                            fprintf(stderr, "Bad
parameter request (q: %d, p: %d)\n",
                                            qnum, vnum);
                        }
              }
    return;
}
```

Appendix C: ACID Scripts

a_query.sql

```
Rem
Rem $Header: a_query.sql 06-aug-99.10:51:10 mpoess Exp $
Rem
Rem a_query.sql
Rem
Rem  Copyright (c) Oracle Corporation 1999. All Rights Reserved.
Rem
Rem    NAME
Rem      a_query.sql - <one-line expansion of the name>
Rem
rem  DESCRIPTION
Rem      Performs ACID Query for TPC-D benchmark.
Rem      Asks user to input values for o_key
Rem      The range of okey is 1 to 600000
Rem
================================================
===============
Rem
Rem Usage:   sqlplus tpcd/tpcd @a_query <o_key>
Rem
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    mpoess      08/06/99 - Creation
Rem    mpoess      08/06/99 - Created
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 = &&1;

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_query2.sql

```
Rem
Rem $Header: aquery2.sql 07-aug-99.23:54:47 mpoess Exp $
Rem
Rem aquery2.sql
Rem
Rem  Copyright (c) Oracle Corporation 1999. All Rights Reserved.
Rem
Rem    NAME
Rem      aquery2.sql - <one-line expansion of the name>
Rem
Rem  DESCRIPTION
Rem      Performs query on PARTSUPP for TPC-D benchmark
Rem      Isolation Test 5.
Rem      Asks user to input values for ps_partkey and ps_suppkey
Rem      The range for ps_partkey is 1 to 20000
Rem      The range for ps_suppkey is 1 to 1000
Rem      A valid combination is 46 and 47
Rem  Usage:   sqlplus tpcd/tpcd @a_query2 <ps_partkey>
<ps_suppkey>
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    mpoess      08/07/99 - Creation
Rem    mpoess      08/07/99 - Created
Rem
rem  DESCRIPTION
rem      Performs query on PARTSUPP for TPC-D benchmark
rem      Isolation Test 5.
rem      Asks user to input values for ps_partkey and ps_suppkey
rem      The range for ps_partkey is 1 to 20000
rem      The range for ps_suppkey is 1 to 1000
rem      A valid combination is 46 and 47

set serverout on;

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

select *
from partsupp
where ps_partkey = &&1
and ps_suppkey = &&2;

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

exit;
```

d_hist.sql

```
Rem
Rem $Header: d_hist.sql 07-aug-99.21:33:08 mpoess Exp $
Rem
Rem d_hist.sql
Rem
Rem  Copyright (c) Oracle Corporation 1999. All Rights Reserved.
Rem
Rem    NAME
Rem      d_hist.sql - <one-line expansion of the name>
Rem
Rem    DESCRIPTION
Rem      Creates a history table for ACID test purpose.
Rem
Rem    NOTES
Rem      <other useful comments, qualifications, etc.>
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    mpoess      08/07/99 - Creation
Rem    mpoess      08/07/99 - Created
Rem


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

drop table history;
```

```
create table history
(
        h_p_key    number,
        h_s_key    number,
        h_o_key number,
        h_l_key    number,
        h_delta number,
        h_date_t date
);

exit;
```

atom.sh

```
#!/bin/ksh
#
# $Header: atom.sh 08-aug-99.13:48:02 mpoess Exp $
#
# atom.sh
#
#  Copyright (c) Oracle Corporation 1999. All Rights Reserved.
#
#    NAME
#      atom.sh - <one-line expansion of the name>
#
#  DESCRIPTION
#      Performs atomicity tests.
#      Usage: atom.sh [-n iter] [-p prog] [-u usr/pswd] -h
#
#      Options: See usage below
#
#    NOTES
#      <other useful comments, qualifications, etc.>
#
#    MODIFIED   (MM/DD/YY)
#    mpoess      08/08/99 - Creation
#    mpoess      08/08/99 - Creation
#

. $KIT_DIR/env

OH=$ORACLE_HOME
# ACID_DIR=$TPCD_KIT_DIR/audit set in env
OUT_DIR=$ACID_OUT
DURA_DIR=$ACID_DIR/dura

usage() {

  echo ""
  echo "Usage: $0 [-n iter] [-p prog] [-u usr/pswd] -h"
  echo ""
  echo "-n iter     : number of iterations, default is 100"
  echo "-p prog     : program to run, default is atranspl.ott"
  echo "-u usr/pswd : user/password combo for database access, default
is tpcd/tpcd"
  echo "-h          : print this usage summary"
  exit 1;
}

ITER=3
SF=1
PROG=atranspl
OUT=${OUT_DIR}/atom
USER=tpcd/tpcd

set -- `getopt "n:p:u:h" "$@"` || usage

while :
do
  case "$1" in
  -n) shift; ITER=$1;;
  -p) shift; PROG=$1;;
  -u) shift; USER=$1;;
  -h) usage; exit 0;;
  --) break;;
  esac
  shift
done

echo "Starting Atomicity Test at `date`..."
echo ""
echo "Performing $ITER ACID transactions with COMMIT"
echo ""

randkey $ITER 1 u$USER | $PROG 1 1 1 0 u$USER > ${OUT}c 2>&1

echo "ACID transactions with COMMIT ended. Output in ${OUT}c"
echo ""
echo "Performing $ITER ACID transactions with ROLLBACK"
echo ""

randkey $ITER 1 u$USER | $PROG 1 1 0 0 u$USER > ${OUT}r 2>&1

echo "ACID transactions with ROLLBACK ended. Output in ${OUT}r"
echo ""
echo "Ending Atomicity Test at `date`..."
```

consist.sh

```
#!/bin/ksh
#
# $Header: consist.sh 08-aug-99.14:20:51 mpoess Exp $
#
# consist.sh
#
#  Copyright (c) Oracle Corporation 1999. All Rights Reserved.
#
#    NAME
#      consist.sh - <one-line expansion of the name>
#
#  DESCRIPTION
#      Performs consistency tests.
#      Usage: consist.sh [-n iter] [-s number of stream] [-p prog]
#                        [-u usr/pswd] -h
#
#      Options: See usage below
#
#    NOTES
#      <other useful comments, qualifications, etc.>
#
#    MODIFIED   (MM/DD/YY)
#    mpoess      08/08/99 - Creation
#    mpoess      08/08/99 - Creation
#

. $KIT_DIR/env

OH=$ORACLE_HOME
# ACID_DIR=$TPCD_KIT_DIR/audit set in env
OUT_DIR=$ACID_OUT

KEY=$OUT_DIR/key$$_
OUTFILE=${OUT_DIR}/consrte
CON1=${OUT_DIR}/conb
CON2=${OUT_DIR}/cona
CHK=${OUT_DIR}/consckpt

/bin/rm -rf ${KEY}* $CON1 $CON2 $OUTFILE $CHK
```

```
trap "/bin/rm -rf ${KEY}*; exit 1" 1 2 3 15

STREAM=2
ITER=100
PROG=atranspl
USER="tpcd/tpcd"
CK=10

usage() {

  echo ""
  echo "Usage: $0 [-n iter] [-s number of stream] [-p prog] [-u
usr/pswd] -h"
  echo ""
  echo "-n iter             : number of iterations, default is 100"
  echo "-s number of stream : number of streams, default is 2"
  echo "-p prog             : program to run, default is atranspl.ott"
  echo "-u usr/pswd         : user/password for database access, default is
tpcd/tpcd"
  echo "-t chkpt            : time after the start of ACID transaction to
perform the checkpoint"
  echo "                      default is 10 seconds"
  echo "-h                  : print this usage summary"
  exit 1;
}

set -- `getopt "n:p:u:s:h" "$@"` || usage

while :
do
  case "$1" in
  -s) shift; STREAM=$1;;
  -n) shift; ITER=$1;;
  -p) shift; PROG=$1;;
  -u) shift; USER=$1;;
  -t) shift; CK=$1;;
  -h) usage; exit 0;;
  --) break;;
  esac
  shift
done


if [ $ITER -lt 100 ]
then
echo "Error: Must at least run 100 iterations!"
echo "Exiting..."
exit 1
fi

if [ $STREAM -lt 2 ]
then
echo "Error: Must at least run 2 streams!"
echo "Exiting..."
exit 1
fi

echo "Starting Consistency Test at `date`..."
echo ""
echo "Generate some keys first"
echo ""

i=0

while [ $i -lt $STREAM ]
do
  echo randkey $ITER 0.1 u$USER
  randkey $ITER 1 u$USER > ${KEY}$i
  i=`expr $i + 1`
done

#exit

echo "Check consistency before Submitting Transactions `date`"
echo "Check consistency before Submitting Transactions `date`" >>
$CON1

echo "Obtain 10 keys from the each key file to check consistency"

i=0
while [ $i -lt $STREAM ]
do
KEYS=`head -10 ${KEY}${i} | awk '{printf "%d ", $1}'`
echo "The 10 Keys for file $i are: $KEYS"
#for j in `head -10 ${KEY}${i} | awk '{printf "%d ", $1}'`
for j in $KEYS
do
   sqlplus $USER @consist $j >> $CON1
   echo "------------------------------" >> $CON1
done
   i=`expr $i + 1`
done

echo ""
echo "Starting ACID transactions at `date`"
echo ""

i=0

while [ $i -lt $STREAM ]
do
   $PROG $i $STREAM 1 0 u${USER} i${KEY}${i}
o${OUTFILE}${i} s1 &
   i=`expr $i + 1`
done

echo "Schedule a Checkpoint"
echo "Checkpoint scheduled at $CK seconds after `date`"

(sleep $CK; $ACID_DIR/consistency/ckpt.sh) &

wait

echo ""
echo "Ending ACID transactions at `date`"
echo ""

echo "Completed $STREAM transaction streams with $ITER iterations
each"
echo ""

echo "Check consistency after Submitting Transactions `date`"
echo "Check consistency after Submitting Transactions `date`" >>
$CON2

cat ${ORACLE_HOME}/rdbms/log/alert_${ORACLE_SID}.log >>
$CHK

i=0
while [ $i -lt $STREAM ]
do
KEYS=`head -10 ${KEY}$i | awk '{printf "%d ", $1}'`
#for j in `head -10 ${KEY}$i | awk '{printf "%d ", $1}'`
echo "The keys to check for consistency after the test from file $i are:"
echo "$KEYS"
for j in $KEYS
do
   sqlplus $USER @consist $j >> $CON2
   echo "------------------------------" >> $CON2
done
   i=`expr $i + 1`
done
```

consist.sql

```
Rem
Rem $Header: consist.sql 08-aug-99.16:59:17 mpoess Exp $
Rem
Rem consist.sql
Rem
Rem  Copyright (c) Oracle Corporation 1999. All Rights Reserved.
Rem
Rem    NAME
Rem      consist.sql - <one-line expansion of the name>
Rem
Rem  DESCRIPTION
Rem       Verifies the consistency of TPC-D database using the
Rem       consistency condition.
Rem
Rem Usage:   sqlplus tpcd/tpcd @consist
Rem
Rem NOTE
Rem REQUIRES PACKAGES prvtotpt and dbmsotpt
rem
Rem    MODIFIED   (MM/DD/YY)
Rem    mpoess      08/08/99 - Creation
Rem    mpoess      08/08/99 - Created
Rem

set verify off
rem set termout on
rem set echo on


REM
REM Get today's date.
REM

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

set serverout on;

DECLARE
              o_okey          number;
              o_tprice     number;
              l_tprice     number;
              diff                        number;
BEGIN
   select o_totalprice
              into o_tprice
              from orders
              where o_orderkey = &&1;


   select sum(trunc((trunc((l_extendedprice * (1-l_discount)), 2)
         * (1+l_tax)), 2))
              into l_tprice
              from lineitem
              where l_orderkey = &&1;

   diff := l_tprice - o_tprice;

   dbms_output.put_line('O_TOTALPRICE: ' ||
TO_CHAR(trunc(o_tprice,2)));
   dbms_output.put_line('L_TOTALPRICE: ' ||
TO_CHAR(trunc(l_tprice,2)));
   dbms_output.put_line('Difference:   ' || TO_CHAR(trunc(diff,2)));

END;
.
/

spool off
exit
```

iso1.sh

```
#!/bin/ksh
#
# $Header: iso.sh 17-aug-99.15:44:51 mpoess Exp $
#
# iso.sh
#
#  Copyright (c) Oracle Corporation 1999. All Rights Reserved.
#
#    NAME
#      iso.sh
#
#    DESCRIPTION
#     This script triggers all 6 isolation tests. In addition,
#     it creates more readable formats of the isolation test output.
#    NOTES
#      <other useful comments, qualifications, etc.>
#
#    MODIFIED   (MM/DD/YY)
#    mpoess      08/17/99 - Creation
#    mpoess      08/17/99 - Creation
#

for iso in iso1 iso2 iso3 iso4 iso5 iso6;do
              echo Running isolation test $iso
              ${iso}.sh
              echo Creating nicely formated output of ACID test $iso
              xiso.pl -o ${ACID_OUT}/${iso}
done
```

iso2.sh

```
#!/bin/ksh
#
# $Header: iso2.sh 04-aug-99.09:19:54 mpoess Exp $
#
# iso2.sh
#
#  Copyright (c) Oracle Corporation 1999. All Rights Reserved.
#
#    NAME
#      iso2.sh - <one-line expansion of the name>
#
#    DESCRIPTION
#      Usage: iso2.sh [-u user/password] [-n remote_node] -h
#      Options: See usage below
#    NOTES
#       For a cross node isolation test, assume the local node is
#       one of the participating nodes.  The other node can be
#       specified by the -n option.
#       You need to set the environment variable TPCD_KIT_DIR
#
#    MODIFIED   (MM/DD/YY)
#    mpoess      08/04/99 - Creation
#    mpoess      08/04/99 - Creation
#
#
#
================================================
===============+
# May need to change the following:

. $KIT_DIR/env
```

```
RSH=rsh

OH=$ORACLE_HOME
# ACID_DIR=$TPCD_KIT_DIR/audit is set in env
OUT_DIR=$ACID_OUT

DURA_DIR=$ACID_DIR/dura

TXN1FILE=$OUT_DIR/txn1$$.out
TXN2FILE=$OUT_DIR/txn2$$.out
KEYFILE=$OUT_DIR/key$$.out
ISOFILE=$OUT_DIR/iso2

USER="tpcd/tpcd"
PROG=atranspl

/bin/rm -rf $TXN1FILE $TXN2FILE $KEYFILE

trap "/bin/rm -rf $TXN1FILE $TXN2FILE $KEYFILE; exit 1" 1 2 3 15

usage() {

  echo ""
  echo "Usage: $0 [-u user/passwd] [-n remote_node] -h"
  echo ""
  exit 1;
}

set -- `getopt "u:n:h" "$@"` || usage

while :
do
  case "$1" in
  -u) shift; USER=$1;;
  -n) shift; HOST="$1";;
  -h) usage; exit 0;;
  --) break;;
  esac
  shift;
done


# generate key files

randkey 1 1 u"$USER" > $KEYFILE

OKEY=`cat $KEYFILE | awk '{print $1}'`
echo "o_key is "$OKEY

# before the ACID transaction, let's run a ACID query to record the
# initial state of lineitem

echo "Running ACID query BEFORE the start of Isolation Test 1" >>
$TXN2FILE
echo "`date`" >> $TXN2FILE
echo "" >> $TXN2FILE
sqlplus $USER @$ACID_DIR/isolation/a_query $OKEY >>
$TXN2FILE
echo "" >> $TXN2FILE
echo "----------------------------------------------------" >> $TXN2FILE

sleep 1

# start ACID transaction, Sleep for 30 second before ROLLBACK

$PROG 1 1 0 0 i$KEYFILE u$USER s30 >> $TXN1FILE &

# let's sleep 10 seconds before starting ACID query

sleep 10

# start ACID query with the same OKEY

echo "Running ACID query 10 seconds AFTER the start of ACID
transaction" \
>> $TXN2FILE
echo "`date`" >> $TXN2FILE
if [ "$HOST" != "" ]
then
echo "Starting ACID query on node $HOST" >> $TXN2FILE
${RSH} -n ${HOST}  sqlplus $USER @$ACID_DIR/isolarion/a_query
$OKEY >> $TXN2FILE
else
sqlplus $USER @$ACID_DIR/isolation/a_query $OKEY >>
$TXN2FILE
fi

echo "----------------------------------------------------" >> $TXN2FILE
wait
echo "----------------------------------------------------" >> $TXN1FILE

cat $TXN1FILE $TXN2FILE >> $ISOFILE

#/bin/rm -rf $TXN1FILE $TXN2FILE $KEYFILE
```

iso3.sh

```
#!/bin/ksh
#
# $Header: iso3.sh 04-aug-99.09:20:35 mpoess Exp $
#
# iso3.sh
#
#  Copyright (c) Oracle Corporation 1999. All Rights Reserved.
#
#    NAME
#      iso3.sh - <one-line expansion of the name>
#
#  DESCRIPTION
#      Usage: iso3.sh [-u user/password] [-n remote_node] -h
#      Options: See usage below
#  NOTES
#       For a cross node isolation test, assume the local node is
#       one of the participating nodes.  The other node can be
#       specified by the -n option.
#              We need to make sure the remote node has access to the
#              file system on the local node.  Otherwise, we need to rcp
#              the keyfile to the remote system.
#       You need to set the environment variable TPCD_KIT_DIR
#
#    MODIFIED   (MM/DD/YY)
#    mpoess      08/04/99 - Creation
#    mpoess      08/04/99 - Creation
#

. $KIT_DIR/env

# May need to change the following:
RSH=rsh

OH=$ORACLE_HOME
#ACID_DIR=$TPCD_KIT_DIR/audit is set in env
OUT_DIR=$ACID_OUT

DURA_DIR=$ACID_DIR/dura

TXN1FILE=$OUT_DIR/txn1$$.out
TXN2FILE=$OUT_DIR/txn2$$.out
KEYFILE=$OUT_DIR/key$$.out
ISOFILE=$OUT_DIR/iso3

USER="tpcd/tpcd"
PROG=atranspl
```

```
/bin/rm -rf $TXN1FILE $TXN2FILE $KEYFILE

trap "/bin/rm -rf $TXN1FILE $TXN2FILE $KEYFILE; exit 1" 1 2 3 15

usage() {

  echo ""
  echo "Usage: $0 [-u user/passwd] [-n remote_node] -h"
  echo ""
  exit 1;
}

set -- `getopt "u:n:h" "$@"` || usage

while :
do
  case "$1" in
  -u) shift; USER=$1;;
  -n) shift; HOST="$1";;
  -h) usage; exit 0;;
  --) break;;
  esac
  shift
done


# generate key files

randkey 1 1 u"$USER" > $KEYFILE

sleep 1

# start ACID transaction, Sleep for 30 second before COMMIT

$PROG 1 2 1 0 i$KEYFILE u$USER s30 >> $TXN1FILE &

# let's sleep 10 seconds before starting second ACID transaction

sleep 10

# start another ACID transaction with the same LKEY and OKEY
# but different DELTA

# Do not sleep before COMMIT so that we can see TXN2 has waited.

if [ "$HOST" != "" ]
then
echo "Starting TXN2 on node $HOST" >> $TXN2FILE
${RSH} -n ${HOST} $PROG 2 2 1 1 i$KEYFILE u$USER s1 >>
$TXN2FILE &
else
$PROG 2 2 1 1 i$KEYFILE u$USER s1 >> $TXN2FILE &
fi

wait
echo "----------------------------------------------------" >> $TXN2FILE
echo "----------------------------------------------------" >> $TXN1FILE

cat $TXN1FILE $TXN2FILE >> $ISOFILE

/bin/rm -rf $TXN1FILE $TXN2FILE $KEYFILE
```

iso4.sh

```
#!/bin/ksh
#
# $Header: iso4.sh 04-aug-99.09:21:12 mpoess Exp $
#
# iso4.sh
#
#  Copyright (c) Oracle Corporation 1999. All Rights Reserved.
#
#    NAME
#      iso4.sh - <one-line expansion of the name>
#
#  DESCRIPTION
#      Usage: iso4.sh [-u user/password] [-n remote_node] -h
#      Options: See usage below
#  NOTES
#       For a cross node isolation test, assume the local node is
#       one of the participating nodes.  The other node can be
#       specified by the -n option.
#       We need to make sure the remote node has access to the
#       file system on the local node.  Otherwise, we need to rcp
#       the keyfile to the remote system.
#       You need to set the environment variable TPCD_KIT_DIR
#
#    MODIFIED   (MM/DD/YY)
#    mpoess      08/04/99 - Creation
#    mpoess      08/04/99 - Creation
#

. $KIT_DIR/env

# May need to change the following:
RSH=rsh

OH=$ORACLE_HOME
# ACID_DIR=$TPCD_KIT_DIR/audit is set in env
OUT_DIR=$ACID_OUT

DURA_DIR=$ACID_DIR/dura

TXN1FILE=$OUT_DIR/txn1$$.out
TXN2FILE=$OUT_DIR/txn2$$.out
KEYFILE=$OUT_DIR/key$$.out
ISOFILE=$OUT_DIR/iso4

USER="tpcd/tpcd"
PROG=atranspl

/bin/rm -rf $TXN1FILE $TXN2FILE $KEYFILE

trap "/bin/rm -rf $TXN1FILE $TXN2FILE $KEYFILE; exit 1" 1 2 3 15

usage() {

  echo ""
  echo "Usage: $0 [-u user/passwd] [-n remote_node] -h"
  echo ""
  exit 1;
}

set -- `getopt "u:n:h" "$@"` || usage

while :
do
  case "$1" in
  -u) shift; USER=$1;;
  -n) shift; HOST="$1";;
  -h) usage; exit 0;;
  --) break;;
  esac
  shift
done


# generate key files

randkey 1 1 u"$USER" > $KEYFILE
```

```
sleep 1

# start ACID transaction, Sleep for 30 second before ROLLBACK

$PROG 1 2 0 0 i$KEYFILE u$USER s30 >> $TXN1FILE &

# let's sleep 10 seconds before starting second ACID transaction

sleep 10

# start another ACID transaction with the same LKEY and OKEY
# but different DELTA

# Do not sleep before COMMIT so that we can see TXN2 has waited.

if [ "$HOST" != "" ]
then
echo "Starting TXN2 on node $HOST" >> $TXN2FILE
${RSH} -n ${HOST} $PROG 2 2 1 1 i$KEYFILE u$USER s1 >>
$TXN2FILE &
else
$PROG 2 2 1 1 i$KEYFILE u$USER s1 >> $TXN2FILE &
fi

wait
echo "------------------------------------------------" >> $TXN2FILE
echo "------------------------------------------------" >> $TXN1FILE

cat $TXN1FILE $TXN2FILE >> $ISOFILE

/bin/rm -rf $TXN1FILE $TXN2FILE $KEYFILE
```

iso5.sh

```
#!/bin/ksh
#
# $Header: iso5.sh 04-aug-99.09:21:45 mpoess Exp $
#
# iso5.sh
#
#  Copyright (c) Oracle Corporation 1999. All Rights Reserved.
#
#    NAME
#      iso5.sh - <one-line expansion of the name>
#
#  DESCRIPTION
#      Usage: iso5.sh [-u user/password] [-n remote_node] -h
#      Options: See usage below
#  NOTES
#       For a cross node isolation test, assume the local node is
#       one of the participating nodes.  The other node can be
#       specified by the -n option.
#       You need to set the environment variable TPCD_KIT_DIR
#
#    MODIFIED   (MM/DD/YY)
#    mpoess      08/04/99 - Creation
#    mpoess      08/04/99 - Creation
#

. $KIT_DIR/env

# May need to change the following:
RSH=rsh

OH=$ORACLE_HOME
# ACID_DIR=$TPCD_KIT_DIR/audit is set in env
OUT_DIR=$ACID_OUT
DURA_DIR=$ACID_DIR/dura

TXN1FILE=$OUT_DIR/txn1$$.out
TXN2FILE=$OUT_DIR/txn2$$.out
KEYFILE=$OUT_DIR/key$$.out
ISOFILE=$OUT_DIR/iso5

USER="tpcd/tpcd"
PROG=atranspl

/bin/rm -rf $TXN1FILE $TXN2FILE $KEYFILE

trap "/bin/rm -rf $TXN1FILE $TXN2FILE $KEYFILE; exit 1" 1 2 3 15

usage() {

  echo ""
  echo "Usage: $0 [-u user/passwd] [-n remote_node] -h"
  echo ""
  exit 1;
}

set -- `getopt "u:n:h" "$@"` || usage

while :
do
  case "$1" in
  -u) shift; USER=$1;;
  -n) shift; HOST="$1";;
  -h) usage; exit 0;;
  --) break;;
  esac
  shift;
done


# generate key files

randkey 1 1 u"$USER" > $KEYFILE

OKEY=`cat $KEYFILE | awk '{print $1}'`
echo "o_key is "$OKEY

# before the ACID transaction, let's run a ACID query to record the
# initial state of lineitem

echo "Running ACID query BEFORE the start of Isolation Test 5" >>
$TXN1FILE
echo "`date`" >> $TXN1FILE
echo "" >> $TXN1FILE
sqlplus $USER @$ACID_DIR/isolation/a_query $OKEY >>
$TXN1FILE
echo "" >> $TXN1FILE
echo "------------------------------------------------" >> $TXN1FILE

sleep 1

# start ACID transaction, Sleep for 60 second before COMMIT

$PROG 1 1 1 0 i$KEYFILE u$USER s60 >> $TXN1FILE &

# let's sleep 5 seconds before starting PARTSUPP query

sleep 5

# First generate PS_PARTKEY and PS_SUPPKEY

PSKEY=`randpsup 0.1`

echo "Running PARTSUPP query 5 seconds AFTER the start of ACID
Transaction" \
>> $TXN2FILE
echo "`date`" >> $TXN2FILE
echo "PS_PARTKEY and PS_SUPPKEY are: $PSKEY" >>
$TXN2FILE
```

```
if [ "$HOST" != "" ]
then
echo "Starting PARTSUPP query on node $HOST" >> $TXN2FILE
${RSH} -n ${HOST} sqlplus $USER
@$ACID_DIR/isolation/a_query2 ${PSKEY} >> $TXN2FILE &
else
sqlplus $USER @$ACID_DIR/isolation/a_query2 ${PSKEY} >>
$TXN2FILE &
fi

wait

echo "-------------------------------------------------" >> $TXN2FILE
echo "-------------------------------------------------" >> $TXN1FILE

cat $TXN1FILE $TXN2FILE >> $ISOFILE

/bin/rm -rf $TXN1FILE $TXN2FILE $KEYFILE
```

iso6.sh

```
#!/bin/ksh
#
# $Header: iso6.sh 04-aug-99.09:22:12 mpoess Exp $
#
# iso6.sh
#
#  Copyright (c) Oracle Corporation 1999. All Rights Reserved.
#
#    NAME
#      iso6.sh - <one-line expansion of the name>
#
#  DESCRIPTION
#      Usage: iso6.sh [-u user/password] [-n remote_node] -h
#      Options: See usage below
#  NOTES
#       For a cross node isolation test, assume the local node is
#       one of the participating nodes.  The other node can be
#       specified by the -n option.
#       We need to make sure the remote node has access to the
#       file system on the local node.  Otherwise, we need to rcp
#       the keyfile to the remote system.
#       You need to set the environment variable TPCD_KIT_DIR
#
#    MODIFIED   (MM/DD/YY)
#    mpoess      08/04/99 - Creation
#    mpoess      08/04/99 - Creation
#

. $KIT_DIR/env

# May need to change the following:
RSH=rsh

OH=/private/tpcd
# ACID_DIR=$TPCD_KIT_DIR/audit is set in env
OUT_DIR=$ACID_OUT

DURA_DIR=$ACID_DIR/dura

TXN1FILE=$OUT_DIR/txn1$$.out
TXN2FILE=$OUT_DIR/txn2$$.out
TXN3FILE=$OUT_DIR/txn3$$.out
KEYFILE=$OUT_DIR/key$$.out
ISOFILE=$OUT_DIR/iso6

USER="tpcd/tpcd"
PROG=atranspl

/bin/rm -rf $TXN1FILE $TXN2FILE $TXN3FILE $KEYFILE

trap "/bin/rm -rf $TXN1FILE $TXN2FILE $TXN3FILE $KEYFILE;
exit 1" 1 2 3 15

usage() {

  echo ""
  echo "Usage: $0 [-u user/passwd] [-n remote_node] -h"
  echo ""
  exit 1;
}

set -- `getopt "u:n:h" "$@"` || usage

while :
do
  case "$1" in
  -u) shift; USER=$1;;
  -n) shift; HOST="$1";;
  -h) usage; exit 0;;
  --) break;;
  esac
  shift;
done


# generate key files

randkey 1 1 u"$USER" > $KEYFILE

OKEY=`cat $KEYFILE | awk '{print $1}'`
echo "o_key is "$OKEY

# before the any transaction, let's run a ACID query to record the
# initial state of lineitem

echo "Running ACID query BEFORE the start of Isolation Test 6" >>
$TXN2FILE
echo "`date`" >> $TXN2FILE
echo "" >> $TXN2FILE
sqlplus $USER @$ACID_DIR/isolation/a_query $OKEY >>
$TXN2FILE
echo "" >> $TXN2FILE
echo "-------------------------------------------------" >> $TXN2FILE

sleep 1

# start Query 17, use 0 as the delta

echo "Running Query 17b at `date`" >> $TXN1FILE
sqlplus $USER @a_q17b >> $TXN1FILE &

# sleep 2 seconds before starting ACID transaction

sleep 2

# start ACID transaction, COMMIT after one second

echo "Starting AICD transaction at `date`" >> $TXN2FILE

if [ "$HOST" != "" ]
then
echo "Starting ACID transaction on node $HOST" >> $TXN2FILE
${RSH} -n ${HOST} $PROG 1 1 1 0 i$KEYFILE u$USER s1 >>
$TXN2FILE &
else
$PROG 1 1 1 0 i$KEYFILE u$USER s1 >> $TXN2FILE &
fi

# start Query 17
```

```
sleep 2

echo "Running 2nd Query 17b at `date`" >> $TXN3FILE
sqlplus $USER @a_q17b >> $TXN3FILE &
# wait for everyone to finish

wait

echo "------------------------------------------------" >> $TXN3FILE
echo "------------------------------------------------" >> $TXN2FILE
echo "------------------------------------------------" >> $TXN1FILE

cat $TXN1FILE $TXN2FILE $TXN3FILE >> $ISOFILE

/bin/rm -rf $TXN1FILE $TXN2FILE $TXN3FILE $KEYFILE
```

iso6_lorna.sh

```
#!/bin/ksh
#
# $Header: iso6.sh 04-aug-99.09:22:12 mpoess Exp $
#
# iso6.sh
#
#  Copyright (c) Oracle Corporation 1999. All Rights Reserved.
#
#   NAME
#     iso6.sh - <one-line expansion of the name>
#
#  DESCRIPTION
#     Usage: iso6.sh [-u user/password] [-n remote_node] -h
#     Options: See usage below
#  NOTES
#      For a cross node isolation test, assume the local node is
#      one of the participating nodes.  The other node can be
#      specified by the -n option.
#      We need to make sure the remote node has access to the
#      file system on the local node.  Otherwise, we need to rcp
#      the keyfile to the remote system.
#      You need to set the environment variable TPCD_KIT_DIR
#
#   MODIFIED   (MM/DD/YY)
#   mpoess      08/04/99 - Creation
#   mpoess      08/04/99 - Creation
#

. $KIT_DIR/env

# May need to change the following:
RSH=rsh

OH=/private/tpcd
# ACID_DIR=$TPCD_KIT_DIR/audit is set in env
OUT_DIR=$ACID_OUT

DURA_DIR=$ACID_DIR/dura

TXN1FILE=$OUT_DIR/txn1$$.out
TXN2FILE=$OUT_DIR/txn2$$.out
TXN3FILE=$OUT_DIR/txn3$$.out
KEYFILE=$OUT_DIR/key$$.out
ISOFILE=$OUT_DIR/iso6

USER="tpcd/tpcd"
PROG=atranspl

/bin/rm -rf $TXN1FILE $TXN2FILE $TXN3FILE $KEYFILE

trap "/bin/rm -rf $TXN1FILE $TXN2FILE $TXN3FILE $KEYFILE;
exit 1" 1 2 3 15

usage() {

  echo ""
  echo "Usage: $0 [-u user/passwd] [-n remote_node] -h"
  echo ""
  exit 1;
}

set -- `getopt "u:n:h" "$@"` || usage

while :
do
  case "$1" in
  -u) shift; USER=$1;;
  -n) shift; HOST="$1";;
  -h) usage; exit 0;;
  --) break;;
  esac
  shift;
done


# generate key files

randkey 1 1 u"$USER" > $KEYFILE

OKEY=`cat $KEYFILE | awk '{print $1}'`
echo "o_key is "$OKEY

# before the any transaction, let's run a ACID query to record the
# initial state of lineitem

echo "Running ACID query BEFORE the start of Isolation Test 6" >>
$TXN2FILE
echo "`date`" >> $TXN2FILE
echo "" >> $TXN2FILE
sqlplus $USER @$ACID_DIR/isolation/a_query $OKEY >>
$TXN2FILE
echo "" >> $TXN2FILE
echo "------------------------------------------------" >> $TXN2FILE

sleep 1

# start Query 17, use 0 as the delta

echo "Running Query 17b at `date`" >> $TXN1FILE
sqlplus $USER @a_q17b >> $TXN1FILE &

# sleep 2 seconds before starting ACID transaction

# followin line deleted by Lorna 9/25/02
#sleep 2

# start ACID transaction, COMMIT after one second

echo "Starting AICD transaction at `date`" >> $TXN2FILE

if [ "$HOST" != "" ]
then
echo "Starting ACID transaction on node $HOST" >> $TXN2FILE
${RSH} -n ${HOST} $PROG 1 1 1 0 i$KEYFILE u$USER s1 >>
$TXN2FILE &
else
$PROG 1 1 1 0 i$KEYFILE u$USER s1 >> $TXN2FILE &
fi

# start Query 17
# following line deleted by Lorna 9/25/02
#sleep 2

echo "Running 2nd Query 17b at `date`" >> $TXN3FILE
sqlplus $USER @a_q17b >> $TXN3FILE &
```

```
## The following lines were repeated by Lorna 9/25/02
sqlplus $USER @a_q17b >> $TXN3FILE
sleep 1
sqlplus $USER @a_q17b >> $TXN3FILE
sleep 1
sqlplus $USER @a_q17b >> $TXN3FILE
sleep 1
sqlplus $USER @a_q17b >> $TXN3FILE
sleep 1
sqlplus $USER @a_q17b >> $TXN3FILE
sleep 1
sqlplus $USER @a_q17b >> $TXN3FILE
sleep 1
sqlplus $USER @a_q17b >> $TXN3FILE
sleep 1
sqlplus $USER @a_q17b >> $TXN3FILE
sleep 1
sqlplus $USER @a_q17b >> $TXN3FILE
sleep 1
sqlplus $USER @a_q17b >> $TXN3FILE
sleep 1
sqlplus $USER @a_q17b >> $TXN3FILE
sleep 1
sqlplus $USER @a_q17b >> $TXN3FILE
sleep 1
sqlplus $USER @a_q17b >> $TXN3FILE
sleep 1
sqlplus $USER @a_q17b >> $TXN3FILE
sleep 1
sqlplus $USER @a_q17b >> $TXN3FILE
sleep 1
sqlplus $USER @a_q17b >> $TXN3FILE
sleep 1
sqlplus $USER @a_q17b >> $TXN3FILE
sleep 1
sqlplus $USER @a_q17b >> $TXN3FILE
sleep 1
sqlplus $USER @a_q17b >> $TXN3FILE
sleep 1
sqlplus $USER @a_q17b >> $TXN3FILE
# wait for everyone to finish

wait

echo "------------------------------------------------" >> $TXN3FILE
echo "------------------------------------------------" >> $TXN2FILE
echo "------------------------------------------------" >> $TXN1FILE

cat $TXN1FILE $TXN2FILE $TXN3FILE >> $ISOFILE

/bin/rm -rf $TXN1FILE $TXN2FILE $TXN3FILE $KEYFILE
```

atranspl.c

```
/* Copyright (c) Oracle Corporation 2001. All Rights Reserved. */

/*

  NAME
    atranspl.c - <one-line expansion of the name>

  DESCRIPTION
    TPC-HR benchmark ACID transaction driver, OCI version 8

  NOTES
    <other useful comments, qualifications, etc.>

  MODIFIED   (MM/DD/YY)
  mpoess      01/04/01 - Creation

*/

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

#include "atranspl.h"

/* Declare error handling functions */

double gettime();
void sql_error();
void usage();
void ACIDinit();
void ACIDexit();
int atoi();
void srand48();
long lrand48();

/* declarations for ORDERS */

int o_key = 0;
double o_tprice = 0.0;
double o_newtprice = 0.0;

/* declarations for LINEITEM */

int l_key = 0;
int l_pkey = 0;
int l_skey = 0;

int l_quan = 0;
int l_newquan = 0;
double l_eprice = 0.0;
double l_neweprice = 0.0;
double l_disc = 0.0;
double l_tax = 0.0;

sb2 l_npricei;

/* other declarations */

int delta = 0;
double rprice;
double cost;

int proc_no = 1;          /* process number, global              */
int num_streams = 1;      /* number of transaction streams       */
int trig = 0;             /* Trigger Time                        */
int slp = 0;              /* Sleep Time                          */

int logfile;              /* fdes for logfile for durability (optional)   */
int outfile = 1;          /* output file (optional)              */
#ifdef LINUX
 FILE *infile;
#else
 FILE *infile = stdin;     /* input file (optional)               */
#endif
                          /* in the format of <o_key> <delta>    */

char lname[UNAME_LEN];    /* username/passwd combo               */
char *passwd;             /* pointer to password                 */

char buf[WRITE_BUF_LEN];  /* buffer to write                     */

unsigned flag = (unsigned) 0;      /* flag to store all sorts of options  */
```

```
#define INFILE  0x01u
#define OUTFILE 0x02u
#define LOGFILE 0x04u
#define COMMIT  0x08u
#define DELTA   0x10u

double tr_end = 0.0;      /* transaction end time              */
double tr_start = 0.0;    /* transaction start time            */

int num_iter = 0;         /* number of iterations              */

time_t curr_time;         /* Current Time                      */

/* OCI handles */

OCIEnv *tpcenv = NULL;
OCIServer *tpcsrv = NULL;
OCIError *errhp = NULL;
OCISvcCtx *tpcsvc = NULL;
OCISession *tpcusr = NULL;
OCIStmt *curi = NULL;
OCIStmt *curr = NULL;
OCIStmt *cure1 = NULL;
OCIStmt *cure2 = NULL;

/* OCI bind handles */

#ifdef NOLKEY
OCIBind *l_keyi_bp = NULL;
OCIBind *o_keyi_bp = NULL;
#endif /* NOLKEY */

OCIBind *l_key_bp = NULL;
OCIBind *o_key_bp = NULL;
OCIBind *delta_bp = NULL;
OCIBind *l_pkey_bp = NULL;
OCIBind *l_skey_bp = NULL;
OCIBind *l_quan_bp = NULL;
OCIBind *l_newquan_bp = NULL;
OCIBind *l_tax_bp = NULL;
OCIBind *l_disc_bp = NULL;
OCIBind *l_eprice_bp = NULL;
OCIBind *l_neweprice_bp = NULL;
OCIBind *o_tprice_bp = NULL;
OCIBind *o_newtprice_bp = NULL;
OCIBind *rprice_bp = NULL;
OCIBind *cost_bp = NULL;

OCIBind *l_neweprice1_bp = NULL;
OCIBind *l_newquan1_bp = NULL;
OCIBind *o_key1_bp = NULL;
OCIBind *l_key1_bp = NULL;

OCIBind *o_newtprice2_bp = NULL;
OCIBind *o_key2_bp = NULL;

sword status = OCI_SUCCESS;  /* OCI return value */

char sqlstmt[1024];

/* usage: prints the usage of the program */

void usage()
{

  fprintf (stderr,"\nUsage: atrans.o[st]t <proc_no> <num_streams>
<commit> <delta>\n[i<pathname for input>] [o<pathname for output>]
[d<pathname for durability file>] [u<uid/passwd>] \n\n");

  fprintf(stderr,"    proc_no      :the process number within this
ACID\n");
  fprintf(stderr,"    num_streams  :the total number of ACID transaction
streams\n");
  fprintf(stderr,"    commit       :1 to commit transaction, abort
otherwise\n\n");
  fprintf(stderr,"    delta        :1 to generate new random delta, otherwise
obtain delta from input\n\n");
  fprintf(stderr,"    OPTIONAL PARAMETERS:\n");
  fprintf(stderr,"    i<pathname for input>      :full path name for input
file - default is stdin\n");
  fprintf(stderr,"    o<pathname for output>     :full path name for output
file - default is stdout\n");
  fprintf(stderr,"    d<pathname for durability> :full path name for
durability success file - must specify for durability test\n");
  fprintf(stderr,"    u<uid/passwd>              :Username/Password string -
default is tcpd/tpcd\n");
  fprintf(stderr,"    t<trigger>                 :Trigger Time - sleep <trigger>
seconds before start\n\n");
  fprintf(stderr,"    s<sleep>                   :Sleep Time - sleep <sleep>
seconds before commit or rollback\n\n");
  exit(-1);

}


void ACIDexit() {

  OCILogoff(tpcsvc,errhp);
  OCIhfree(tpcenv,OCI_HTYPE_STMT);
  OCIhfree(tpcsvc,OCI_HTYPE_SVCCTX);
  OCIhfree(tpcsrv,OCI_HTYPE_SERVER);
  OCIhfree(tpcusr,OCI_HTYPE_SESSION);

}



/* type: 0 if environment handle is passed, 1 if error handle is passwd */

void sql_error(errhp,status,type)
     OCIError *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) OCIErrorGet(errhp,1,NULL,(sb4*) &errcode, (text*) msg,
                         2048, OCI_HTYPE_ERROR);
    else
      (void) OCIErrorGet(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) OCIErrorGet(errhp,1,NULL, (sb4 *) &errcode, (text*) msg,
                         2048,OCI_HTYPE_ERROR);
    else
      (void) OCIErrorGet(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) OCIErrorGet(errhp,1,NULL, (sb4 *) &errcode, (text*) msg,
```

```
                2048,OCI_HTYPE_ERROR);
  else
    (void) OCIErrorGet(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);

 ACIDexit();

 exit(1);
}

#ifdef LINUX
int main(argc,argv)
#else
void main(argc,argv)
#endif
    int argc;
    char *argv[];
{

 int i;
 char line[64];
 ub4 errcode;
 char msg[2048];
 int need_commit = 0;

 /* Initialize some variables */
#ifdef LINUX
 infile=fopen("/dev/stdin","r");
#endif
 strcpy((char *) lname, "tpcd/tpcd");

 if ((argc > 10) || (argc < 5)) {
  usage();
 }

 /* argv[1] -- Process Number */

 proc_no = atoi(argv[1]);

 /* argv[2] -- Number of Streams */

 num_streams = atoi(argv[2]);

 /* argv[3] -- Commit? */

 if (atoi(argv[3]) == 1)
  BIS(flag, COMMIT);

 /* argv[4] -- Delta? */

 if (atoi(argv[4]) == 1)
  BIS(flag, DELTA);

 /* Process  optional parameters */

 argc -= 4;
 argv += 4;

 while(--argc) {
  ++argv;
  switch(argv[0][0]) {
  case 'u':
    strncpy((char *) lname, ++(argv[0]), UNAME_LEN);
    if (strchr((char *) lname, '/') == NULL) {
      fprintf(stderr, "Login name must be in the format of
userid/passwd\n");
      usage();
      exit(-1);
    }
    break;
  case 'i':
    if ((infile = fopen(++(argv[0]), "r")) == NULL) {
      fprintf(stderr,"Cannot open input file %s\n", argv[0]);
      fprintf(stderr,"%s\n",strerror(errno));
      exit(-1);
    }
    BIS(flag, INFILE);
    break;
  case 'o':
    if ((outfile = open(++(argv[0]), (O_RDWR | O_SYNC | O_CREAT),
S_IRWXU)) == -1) {
      fprintf(stderr,"Cannot open output file %s\n", argv[0]);
      fprintf(stderr,"%s\n",strerror(errno));
      exit(-1);
    }
    BIS(flag, OUTFILE);
    break;
  case 'd':
    if ((logfile = open(++(argv[0]), (O_RDWR | O_SYNC | O_CREAT),
S_IRWXU)) == -1) {
      fprintf(stderr,"Cannot open durability success file %s\n", argv[0]);
      fprintf(stderr,"%s\n",strerror(errno));
      exit(-1);
    }
    BIS(flag, LOGFILE);
    break;
  case 't':
    trig = atoi(++(argv[0]));
    break;
  case 's':
    slp = atoi(++(argv[0]));
    break;
  default:
    fprintf(stderr, "Unknown argument %s\n", argv[0]);
    usage();
    break;
  }
 }

 FPRTF(outfile,"-----------------------------------------------\n");

 /* Initialize the cursors etc. */

 (void) ACIDinit();

 /* sleep for some time (triggering) */

 sleep(trig);

 /* start doing the ACID transactions */

 tr_start = gettime();

 /* The number of iteration we will run depends on the number of */
 /* input lines                                                  */

 while (fgets(line, 64, infile) != NULL) {

#ifdef NOLKEY
   sscanf(line, "%d %d\n", &o_key, &delta);

   /* Obtain l_key from l_key query */
```

```
    OCIsexec(tpcsvc,curi,errhp,1);

    /* l_key is the highest l_linenumber available.  We need to pick */
    /* at random a number between 1..l_key.                          */

    l_key = (int) ((lrand48() % l_key) + 1);
#else
    sscanf(line, "%d %d %d\n", &o_key, &l_key, &delta);
#endif /* NOLKEY */

    /* Generate delta if necessary */

    if (BIT(flag, DELTA))
      delta = (int) (floor((drand48() * 100)) + 1);

    /* Now, we are ready to run the ACID transaction. */

    curr_time = time(NULL);

    FPRTF2(outfile, "Starting ACID transaction %d at %s...\n",
(++num_iter),
            ctime(&curr_time));

    FPRTF1(outfile, "o_key: %d\n", (int) o_key);
    FPRTF1(outfile, "l_key: %d\n", (int) l_key);
    FPRTF1(outfile, "delta: %d\n", (int) delta);

    OCIsexec(tpcsvc,curr,errhp,1);

           /*sql_error(errhp,status,1); */
    curr_time = time(NULL);

    if (!BIT(flag, LOGFILE)) {
    FPRTF1(outfile, "BEFORE COMMIT/ROLLBACK
TRANSACTION at %s\n", ctime(&curr_time));
    FPRTF1(outfile, "l_extendedprice: %.2f\n", l_eprice);
    FPRTF1(outfile, "l_quantity:      %d\n", (int) l_quan);
    FPRTF1(outfile, "o_totalprice:    %.2f\n\n", o_tprice);
    }

    FPRTF1(outfile, "Sleep %d seconds before
COMMIT/ROLLBACK...\n\n", slp);
    sleep(slp);

    /* Shall we commit? */

    if (BIT(flag, COMMIT)) {
      need_commit = 1;
      while (need_commit) {
        if((status=OCITransCommit(tpcsvc
,errhp,OCI_DEFAULT)) != OCI_SUCCESS) {
                            OCIrol(tpcsvc,errhp);
              OCIsexec(tpcsvc,curr,errhp,1);
        } else {
            need_commit = 0;
            curr_time = time(NULL);
            FPRTF2(outfile, "ACID Transaction iteration %d COMMITED
at %s\n",
              num_iter, ctime(&curr_time));
        }
      }
    } else {
      OCIrol(tpcsvc,errhp);
      curr_time = time(NULL);
      FPRTF2(outfile, "ACID Transaction iteration %d ROLLBACK at
%s\n",
            num_iter, ctime(&curr_time));
    }

    /* Report all results to outfile and if necessary, to success file. */

    /* Report initial and new values for o_totalprice, l_extendedprice, */
    /* l_quantity.                                                       */
    /*
    curr_time = time(NULL);
    FPRTF1(outfile, "Transaction Completed at %s\n",
ctime(&curr_time));
    */

    /* Get the values in LINEITEM and ORDERS after the transaction */

    if (BIT(flag, LOGFILE)) {
      FPRTF1(logfile, "p_key:      %d\n", (int) l_pkey);
      FPRTF1(logfile, "s_key:      %d\n", (int) l_skey);
      FPRTF1(logfile, "o_key:      %d\n", (int) o_key);
      FPRTF1(logfile, "l_key:      %d\n", (int) l_key);
      FPRTF1(logfile, "delta:      %d\n", (int) delta);
      FPRTF1(logfile, "Transaction Completed at %s\n",
ctime(&curr_time));
      FPRTF(logfile, "-------------------------------------------------\n");
    } else {

      OCIsexec(tpcsvc,cure1,errhp,1);
      OCIsexec(tpcsvc,cure2,errhp,1);

      FPRTF(outfile, "AFTER TRANSACTION:\n");
      FPRTF1(outfile, "l_extendedprice: %.2lf\n", l_neweprice);
      FPRTF1(outfile, "l_quantity:      %d\n", (int) l_newquan);
      FPRTF1(outfile, "o_totalprice:    %.2lf\n\n", o_newtprice);
      FPRTF1(outfile, "l_tax:           %.2lf\n", l_tax);
      FPRTF1(outfile, "l_discount:      %.2lf\n", l_disc);
      FPRTF1(outfile, "rprice:          %.2lf\n", rprice);
      FPRTF1(outfile, "cost:            %.2lf\n", cost);
      FPRTF(outfile, "-------------------------------------------------\n");
    }
  }

  tr_end = gettime();

  if (!BIT(flag,LOGFILE)) {
    FPRTF1(outfile, "Start Time: %.2f\n", tr_start);
    FPRTF1(outfile, "End Time: %.2f\n", tr_end);
    FPRTF1(outfile, "Elapsed Time: %.2f\n", (tr_end - tr_start));
    FPRTF1(outfile, "Transaction Count: %d\n", num_iter);
    FPRTF1(outfile, "Transaction Rate: %.2f\n", num_iter/(tr_end -
tr_start));
  } else {
    FPRTF1(logfile, "Start Time: %.2f\n", tr_start);
    FPRTF1(logfile, "End Time: %.2f\n", tr_end);
    FPRTF1(logfile, "Elapsed Time: %.2f\n", (tr_end - tr_start));
    FPRTF1(logfile, "Transaction Count: %d\n", num_iter);
  }

  /*  Disconnect from ORACLE. */

  if (BIT(flag, INFILE))
    fclose(infile);
  if (BIT(flag, OUTFILE))
    close(outfile);
  if (BIT(flag, LOGFILE))
    close(logfile);

  ACIDexit();

  exit(0);
}


void ACIDinit()
{

  /* run random seed */

  srand48(getpid());
```

```
  /* Connect to ORACLE.  Program will call sql_error()
     if an error occurs in connecting to the default database. */

  (void) OCIInitialize(OCI_DEFAULT,(dvoid *)0,0,0,0);
  if((status=OCIEnvInit((OCIEnv **)&tpcenv,OCI_DEFAULT,0,(dvoid
**)0)) !=
     OCI_SUCCESS)
    sql_error(tpcenv, status, 0);

  OCIhalloc(tpcenv,&errhp,OCI_HTYPE_ERROR);
  OCIhalloc(tpcenv,&curi,OCI_HTYPE_STMT);
  OCIhalloc(tpcenv,&curr,OCI_HTYPE_STMT);
  OCIhalloc(tpcenv,&cure1,OCI_HTYPE_STMT);
  OCIhalloc(tpcenv,&cure2,OCI_HTYPE_STMT);
  OCIhalloc(tpcenv,&tpcsvc,OCI_HTYPE_SVCCTX);
  OCIhalloc(tpcenv,&tpcsrv,OCI_HTYPE_SERVER);
  OCIhalloc(tpcenv,&tpcusr,OCI_HTYPE_SESSION);

  /* Disables auto commit */
/*
  if (ocof(&tpclda)) {
    sql_error(&tpclda, &tpclda);
    ologof(&tpclda);
    exit(-1);
  }
*/

  /* get username and password */

  passwd = strchr(lname, '/');
  *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(tpcusr,OCI_HTYPE_SESSION,lname,strlen(lname),OCI_ATT
R_USERNAME,
         errhp);

OCIaset(tpcusr,OCI_HTYPE_SESSION,passwd,strlen(passwd),OCI_A
TTR_PASSWORD,
         errhp);
  if ((status = OCISessionBegin(tpcsvc, errhp, tpcusr,
OCI_CRED_RDBMS,
                        OCI_DEFAULT)) != OCI_SUCCESS)
/*
 if((status =  OCILogon(tpcenv, errhp, &tpcsvc, "tpcd", strlen("tpcd"),
"tpcd", strlen("tpcd"), 0, 0)) != OCI_SUCCESS)
*/
    sql_error(errhp,status,1);


OCIaset(tpcsvc,OCI_HTYPE_SVCCTX,tpcusr,0,OCI_ATTR_SESSIO
N,errhp);

  /* Make session serializable */

  sprintf ((char *) sqlstmt, ISOTXT);
  OCIStmtPrepare(curi,errhp,(text *)sqlstmt,strlen((char *)sqlstmt),
              OCI_NTV_SYNTAX,OCI_DEFAULT);
  OCIsexec(tpcsvc,curi,errhp,1);

  curr_time = time(NULL);
  printf("\nConnected to ORACLE as user: %s at %s\n\n", lname,
ctime(&curr_time));

#ifdef NOLKEY
  /* Open and Parse cursor for query to choose determine l_key. */
  /* Binds l_key to :l_key.                                     */

  sprintf((char *) sqlstmt,SQLTXT1);
  OCIStmtPrepare(curi,errhp,sqlstmt,strlen((char
*)sqlstmt),OCI_NTV_SYNTAX,OCI_DEFAULT);



OCIbbname(curi,&l_keyi_bp,errhp,":l_key",ADR(l_key),SIZ(l_key),SQ
LT_INT);

OCIbbname(curi,&o_keyi_bp,errhp,":o_key",ADR(o_key),SIZ(o_key),S
QLT_INT);

#endif /* NOLKEY */

  /* Open and Parse cursor for the ACID transaction.            */

  sprintf((char *) sqlstmt,SQLTXT2);
  OCIStmtPrepare(curr,errhp,(text *)sqlstmt,strlen((char *)sqlstmt),
              OCI_NTV_SYNTAX,OCI_DEFAULT);

  /* bind variables */


OCIbbname(curr,l_key_bp,errhp,":l_key",ADR(l_key),SIZ(l_key),SQLT
_INT);

OCIbbname(curr,o_key_bp,errhp,":o_key",ADR(o_key),SIZ(o_key),SQ
LT_INT);

OCIbbname(curr,delta_bp,errhp,":delta",ADR(delta),SIZ(delta),SQLT_I
NT);

OCIbbname(curr,l_pkey_bp,errhp,":l_pkey",ADR(l_pkey),SIZ(l_pkey),
SQLT_INT);

OCIbbname(curr,l_skey_bp,errhp,":l_skey",ADR(l_skey),SIZ(l_skey),S
QLT_INT);

OCIbbname(curr,l_quan_bp,errhp,":l_quan",ADR(l_quan),SIZ(l_quan),
SQLT_INT);

OCIbbname(curr,l_newquan_bp,errhp,":l_newquan",ADR(l_newquan),
         SIZ(l_newquan),SQLT_INT);

OCIbbname(curr,l_tax_bp,errhp,":l_tax",ADR(l_tax),SIZ(l_tax),SQLT_
FLT);

OCIbbname(curr,l_disc_bp,errhp,":l_disc",ADR(l_disc),SIZ(l_disc),SQ
LT_FLT);

OCIbbname(curr,l_eprice_bp,errhp,":l_eprice",ADR(l_eprice),SIZ(l_epr
ice),
         SQLT_FLT);

OCIbbname(curr,l_neweprice_bp,errhp,":l_neweprice",ADR(l_newepric
e),
         SIZ(l_neweprice),SQLT_FLT);


OCIbbname(curr,o_tprice_bp,errhp,":o_tprice",ADR(o_tprice),SIZ(o_tp
rice),
         SQLT_FLT);

OCIbbname(curr,o_newtprice_bp,errhp,":o_newtprice",ADR(o_newtpri
ce),
```

```
        SIZ(o_newtprice), SQLT_FLT);
  OCIbbname(curr,rprice_bp,errhp,":rprice",ADR(rprice),SIZ(rprice),
SQLT_FLT);
  OCIbbname(curr,cost_bp,errhp,":cost",ADR(cost),SIZ(cost),
SQLT_FLT);

  /* Open & Parse cursor for end values query */

  sprintf((char *) sqlstmt,SQLTXT3);
  OCIStmtPrepare(cure1,errhp,(text *)sqlstmt,strlen((char *)sqlstmt),
            OCI_NTV_SYNTAX,OCI_DEFAULT);

  sprintf((char *) sqlstmt,SQLTXT4);
  OCIStmtPrepare(cure2,errhp,(text *)sqlstmt,strlen((char *)sqlstmt),
            OCI_NTV_SYNTAX,OCI_DEFAULT);

  /* bind variables */


OCIbbname(cure1,l_neweprice1_bp,errhp,":l_neweprice",ADR(l_newe
price),
        SIZ(l_neweprice),SQLT_FLT);

OCIbbname(cure1,l_newquan1_bp,errhp,":l_newquan",ADR(l_newqua
n),
        SIZ(l_newquan),SQLT_INT);

OCIbbname(cure1,o_key1_bp,errhp,":o_key",ADR(o_key),SIZ(o_key),
SQLT_INT);

OCIbbname(cure1,l_key1_bp,errhp,":l_key",ADR(l_key),SIZ(l_key),SQ
LT_INT);


OCIbbname(cure2,o_newtprice2_bp,errhp,":o_newtprice",ADR(o_newt
price),
        SIZ(o_newtprice),SQLT_FLT);

OCIbbname(cure2,o_key2_bp,errhp,":o_key",ADR(o_key),SIZ(o_key),
SQLT_INT);

}
```

atranspl.h

```
/* Copyright (c) Oracle Corporation 2001. All Rights Reserved. */

/*
  NAME
    atranspl.h - <one-line expansion of the name>

  DESCRIPTION

  MODIFIED   (MM/DD/YY)
  mpoess      01/04/01 - Creation

*/

#ifndef ATRANSPL_H
#define ATRANSPL_H

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/param.h>
#include <sys/types.h>
#include <time.h>
#include <errno.h>
#include <math.h>

#include <oratypes.h>

#ifndef OCIDFN
#include <ocidfn.h>
#endif /* OCIDFN */

#ifndef OCI_ORACLE
#include <oci.h>
#endif /* OCI_ORACLE */

/*
#ifdef __STDC__
#include <ociapr.h>
#else
#include <ocikpr.h>
#endif */ /* __STDC__ */

extern int errno;

#ifndef NULL
#define NULL 0
#endif

#ifndef NULLP
# define NULLP (void *)NULL
#endif /* NULLP */

#ifndef DISCARD
# define DISCARD (void)
#endif

#ifndef sword
# define sword int
#endif

#ifndef ub1
#define ub1 unsigned char
#endif

#define UNAME_LEN 64
#define WRITE_BUF_LEN 1024

#define NA              -1      /* ANSI SQL NULL */
#define VER7            2
#define NOT_SERIALIZABLE  8177  /* ORA-08177: transaction not
serializable */
#define WRITE_BUF_LEN 1024

#define ADR(object) ((ub1 *)&(object))
#define SIZ(object) ((sword)sizeof(object))
#define BIS(flg,mask) (unsigned) (flg |= (unsigned) mask)
#define BIT(flg,mask) (unsigned) ((unsigned) flg & (unsigned) mask)

#define FPRTF(fd,s) \
{sprintf(buf,s); write(fd, buf, strlen(s));}
#define FPRTF1(fd,s,p) \
{sprintf(buf,s,p); write(fd, buf, strlen(buf));}
#define FPRTF2(fd,s,p1,p2) \
{sprintf(buf,s,p1,p2); write(fd, buf, strlen(buf));}

#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 OCIaget(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 OCIaset(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,errh,iter,0,NULL,NULL,OCI_DE
FAULT)) != OCI_SUCCESS) \
      sql_error(errh,status,1); \
   else \
     DISCARD 0

#define OCIbbname(stmh,bindp,errh,sqlvar,progv,progvl,ftype) \
   if((status=OCIBindByName(stmh,&bindp,errh,(text
*)sqlvar,strlen(sqlvar), \
            progv,progvl,ftype,0,0,0,0,0,OCI_DEFAULT)) !=
OCI_SUCCESS) \
      sql_error(errh,status,1); \
   else \
     DISCARD 0

#define OCIbbnamei(stmh,bindp,errh,sqlvar,progv,progvl,ftype,indp) \
   if((status=OCIHandleAlloc((dvoid *)stmh,(dvoid
**)&bindp,OCI_HTYPE_BIND, \
                        0,(dvoid **)0))!=OCI_SUCCESS) \
      sql_error(stmh,status,0); \
   if((status=OCIBindByName(stmh,&bindp,errh,(text
*)sqlvar,strlen(sqlvar), \
            progv,progvl,ftype,indp,0,0,0,0,OCI_DEFAULT)) !=
OCI_SUCCESS) \
      sql_error(errh,status,1); \
   else \
     DISCARD 0

#define OCIcom(svcp,errh) \
    if((status=OCITransCommit(svcp,errh,OCI_DEFAULT)) !=
OCI_SUCCESS) \
       sql_error(errh,status,1); \
    else \
      DISCARD 0

#define OCIrol(svcp,errh) \
    if((status=OCITransRollback(svcp,errh,OCI_DEFAULT)) !=
OCI_SUCCESS) \
       sql_error(errh,status,1); \
    else \
      DISCARD 0

#define ISOTXT "alter session set isolation_level = serializable"

#define SQLTXT1 "BEGIN SELECT MAX(l_linenumber) INTO :l_key
FROM lineitem \
WHERE l_orderkey = :o_key; END;"

#define SQLTXT2 "BEGIN d_atrans.doatrans(:l_key, :o_key, :delta,
:l_pkey, \
:l_skey, :l_quan, :l_newquan, :l_tax, :l_disc, :l_eprice, :l_neweprice, \
:o_tprice, :o_newtprice, :rprice, :cost); END;"

#define SQLTXT3 "BEGIN SELECT l_extendedprice, l_quantity \
INTO :l_neweprice, :l_newquan \
FROM lineitem \
WHERE l_orderkey = :o_key \
AND l_linenumber = :l_key; END;"

#define SQLTXT4 "BEGIN SELECT o_totalprice INTO :o_newtprice \
FROM orders \
WHERE o_orderkey = :o_key; END;"

#define SQLTXT5 "BEGIN SELECT l_extendedprice, l_quantity \
INTO :l_eprice, :l_quan \
FROM lineitem \
WHERE l_orderkey = :o_key \
AND l_linenumber = :l_key; END;"

#define SQLTXT6 "BEGIN SELECT o_totalprice INTO :o_tprice \
FROM orders \
WHERE o_orderkey = :o_key; END;"

#endif /* ATRANSPL_H */
```

atrans.sql

```
Rem
Rem $Header: atrans.sql 07-aug-99.21:27:13 mpoess Exp $
Rem
Rem atrans.sql
Rem
Rem  Copyright (c) Oracle Corporation 1999. All Rights Reserved.
Rem
Rem    NAME
Rem      atrans.sql - <one-line expansion of the name>
Rem
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
Rem    NOTES
Rem      <other useful comments, qualifications, etc.>
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    mpoess      08/07/99 - Creation
Rem    mpoess      08/07/99 - Created
Rem


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

CREATE OR REPLACE PACKAGE d_atrans
IS
PROCEDURE doatrans
(
              l_key                    IN OUT integer,
              o_key                    IN OUT integer,
              delta                    IN OUT integer,
              l_pkey         IN OUT integer,
              l_skey         IN OUT integer,
              l_quan         IN OUT integer,
              l_newquan      IN OUT integer,
              l_tax          IN OUT number,
              l_disc         IN OUT number,
              l_eprice       IN OUT number,
              l_neweprice    IN OUT number,
              o_tprice    IN OUT number,
              o_newtprice              IN OUT number,
              rprice                   IN OUT number,
              cost                     IN OUT number
);
END;
/

CREATE OR REPLACE PACKAGE BODY d_atrans
```

```
IS
PROCEDURE doatrans
(
        l_key                   IN OUT integer,
        o_key                   IN OUT integer,
        delta                   IN OUT integer,
        l_pkey          IN OUT integer,
        l_skey          IN OUT integer,
        l_quan          IN OUT integer,
        l_newquan       IN OUT integer,
        l_tax           IN OUT number,
        l_disc          IN OUT number,
        l_eprice        IN OUT number,
        l_neweprice     IN OUT number,
        o_tprice    IN OUT number,
        o_newtprice             IN OUT number,
        rprice                  IN OUT number,
        cost                    IN OUT number
)
IS
    ototal number;
    not_serializable EXCEPTION;
    PRAGMA EXCEPTION_INIT(not_serializable,-8177);
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_quan, l_eprice, l_pkey, l_skey, l_tax, l_disc
    from lineitem
    where l_orderkey = o_key
    and   l_linenumber = l_key;

  ototal := o_tprice - trunc((trunc((l_eprice * (1.0-l_disc)),2) *
(1.0+l_tax)),2);
  rprice := trunc((l_eprice/l_quan), 2);
  cost := trunc((rprice * delta), 2);
  l_neweprice := l_eprice + cost;
  o_newtprice := trunc((l_neweprice * (1.0 - l_disc)), 2);
  o_newtprice := ototal + trunc((o_newtprice * (1.0 + l_tax)), 2);
  l_newquan := l_quan + delta;

  update lineitem
    set l_extendedprice = l_neweprice,
    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;
 END;

 END LOOP;

END doatrans;
END;
/
exit;
```

randkey.c

```
/* Copyright (c) Oracle Corporation 2001. All Rights Reserved. */

/*

  NAME
    randkey.c - <one-line expansion of the name>

  DESCRIPTION
    Generate random keys for ACID transactions:
    O_ORDERKEY unique random (1..SF*150000*4) and only
    first 8 keys out of every 32 are populated.
      and
    L_ORDERKEY based on Clause 3.1.6.2
    DELTA random (1..100)
*/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "atranspl.h"

#define ORDERCNT 150000.0

/* MK_SPARSE adopted from dss.h */

#define  MK_SPARSE(key, seq) \
        (((((key>>3)<<2)|(seq & 0x0003))<<3)|(key & 0x0007))

void sql_error();
void usage();
void ACIDinit();
long atol();
void srand48();
long lrand48();

/* Not really used here, but retained it for future purposes. */

typedef struct aciddef {
  long okey;
  long lkey;
  int delta;
} adef;

long l_key = 0;
long o_key = 0;
char lname[UNAME_LEN];
char *passwd;

/* OCI handles */

OCIEnv *tpcenv;
OCIServer *tpcsrv;
OCIError *errhp;
OCISvcCtx *tpcsvc;
OCISession *tpcusr;
OCIStmt *curi;

OCIBind *l_key_bp;
OCIBind *o_key_bp;

sword status = OCI_SUCCESS;  /* OCI return value */

char sqlstmt[1024];


void ACIDexit() {
```

```
  OCILogoff(tpcsvc,errhp);
  OCIhfree(tpcenv,OCI_HTYPE_STMT);
  OCIhfree(tpcsvc,OCI_HTYPE_SVCCTX);
  OCIhfree(tpcsrv,OCI_HTYPE_SERVER);
  OCIhfree(tpcusr,OCI_HTYPE_SESSION);
}


/* type: 0 if environment handle is passed, 1 if error handle is passwd */

void sql_error(errhp,status,type)
     OCIError *errhp;
     sword status;
     sword type;
{
  char msg[2048];
  sb4 errcode;
  ub4 msglen;
  int i,j;

  switch(status) {
  case OCI_SUCCESS_WITH_INFO:
    fprintf(stderr, "Error: Statement returned with info.\n");
    if (type)
      (void) OCIErrorGet(errhp,1,NULL,(sb4 *) &errcode,(text *)msg,
                         2048,OCI_HTYPE_ERROR);
    else
      (void) OCIErrorGet(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) OCIErrorGet(errhp,1,NULL,(sb4 *) &errcode,(text *)msg,
                         2048,OCI_HTYPE_ERROR);
    else
      (void) OCIErrorGet(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) OCIErrorGet(errhp,1,NULL,(sb4 *) &errcode,(text *)msg,
                         2048,OCI_HTYPE_ERROR);
    else
      (void) OCIErrorGet(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);

  ACIDexit();

  exit(1);
}


main(argc, argv)
     int argc;
     char **argv;
{

  long count;
  long i;
  double sf;        /* need to accomodate sf 0.1 */
  double random;
  double ordcnt;
  adef *res;

  if ((argc < 3) || (argc > 4)) {
    usage();
    exit(-1);
  }

  strcpy((char *) lname, "tpcd/tpcd");

  count = atol(argv[1]);
  sf = atof(argv[2]);

  argc -= 2;
  argv += 2;

  while (--argc) {
    ++argv;
    switch(argv[0][0]) {
    case 'u':
      strncpy((char *) lname, ++(argv[0]), UNAME_LEN);
      if (strchr((char *) lname, '/') == NULL) {
        usage();
        exit(-1);
      }
      break;
    default:
      fprintf(stderr, "Unknown argument %s\n", argv[0]);
      usage();
      break;
    }
  }

  ACIDinit();

  /* initialize array for random numbers */

  res = (adef *) malloc(count*sizeof(adef));
  ordcnt = (double) ORDERCNT * (double) sf;

  for (i=0; i<count; i++) {

    /* The algorithm:                                          */
    /* Assumes drand's output is 'unique', first get a number within */
    /* the range of [0..sf*ORDERCNT) and then maps the different     */
    /* ranges to generate the real output.                     */

    random = floor(drand48() * (double) ordcnt) + 1;
    res[i].okey = o_key = (long) MK_SPARSE((long) random, 0);
    res[i].delta = (long) floor(drand48() * 100) + 1;

    /* Obtain l_key from l_key query */

    OCIsexec(tpcsvc,curi,errhp,1);

    /* l_key is the highest l_linenumber available.  We need to pick */
    /* at random a number between 1..l_key.                    */

    res[i].lkey = (lrand48() % l_key) + 1;

    printf("%ld %ld %d\n", res[i].okey, res[i].lkey, res[i].delta);
  }

  ACIDexit();
  free(res);

}
```

```
void usage() {

  fprintf(stderr, "Usage: randkey <number of random keys to generate>
<SF> u<user/password>\n");
  fprintf(stderr, "\n");
}

void ACIDinit()
{

  /* run random seed */

  srand48(getpid());

  /* Connect to ORACLE.  Program will call sql_error()
     if an error occurs in connecting to the default database. */

  (void) OCIInitialize(OCI_DEFAULT,(dvoid *)0,0,0,0);
  if((status=OCIEnvInit((OCIEnv **)&tpcenv,OCI_DEFAULT,0,(dvoid
**)0)) !=
     OCI_SUCCESS)
    sql_error(tpcenv, status, 0);

  OCIhalloc(tpcenv,&errhp,OCI_HTYPE_ERROR);
  OCIhalloc(tpcenv,&curi,OCI_HTYPE_STMT);
  OCIhalloc(tpcenv,&tpcsvc,OCI_HTYPE_SVCCTX);
  OCIhalloc(tpcenv,&tpcsrv,OCI_HTYPE_SERVER);
  OCIhalloc(tpcenv,&tpcusr,OCI_HTYPE_SESSION);

  /* get username and password */

  passwd = strchr(lname, '/');
  *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(tpcusr,OCI_HTYPE_SESSION,lname,strlen(lname),OCI_ATT
R_USERNAME,
         errhp);

OCIaset(tpcusr,OCI_HTYPE_SESSION,passwd,strlen(passwd),OCI_A
TTR_PASSWORD,
         errhp);

  if ((status = OCISessionBegin(tpcsvc, errhp, tpcusr,
OCI_CRED_RDBMS,
                        OCI_DEFAULT)) != OCI_SUCCESS)
    sql_error(errhp,status,1);


OCIaset(tpcsvc,OCI_HTYPE_SVCCTX,tpcusr,0,OCI_ATTR_SESSIO
N,errhp);

  /* Open and Parse cursor for query to choose determine l_key. */
  /* Binds l_key to :l_key.                                     */

  sprintf((char *) sqlstmt,SQLTXT1);
  OCIStmtPrepare(curi,errhp,(text *)sqlstmt,strlen((char *)sqlstmt),
              OCI_NTV_SYNTAX,OCI_DEFAULT);


OCIbbname(curi,l_key_bp,errhp,":l_key",ADR(l_key),SIZ(l_key),SQLT
_INT);

OCIbbname(curi,o_key_bp,errhp,":o_key",ADR(o_key),SIZ(o_key),SQ
LT_INT);
}
```

randpsup.c

```
/* Copyright (c) Oracle Corporation 2001. All Rights Reserved. */

/*

   NAME
     randpsup.c - <one-line expansion of the name>

   DESCRIPTION
     Generate random keys for ACID PARTSUPP transactions:
     (Clause 4.2.3)
     PS_PARTKEY random within [SF*200000]
       and
     PS_SUPPKEY = (PS_PARTKEY + (i * ((S/4) +
(int)(PS_PARTKEY - 1)
                /S))) % S + 1
     where i random within [0..3] and S = SF * 10000

MODIFIED
   mpoess      01/04/01 - Creation

*/

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

#define PS_PER_SF 200000.0
#define S_PER_SF 10000.0
#define SUPP_PER_PART 4

/* borrowed from build.c in the dbgen distribution */

#define PART_SUPP_BRIDGE(tgt, p, s) \
    { \
    long tot_scnt = (long) (S_PER_SF * sf); \
    tgt = (p + s *  (tot_scnt / SUPP_PER_PART +  \
       (long) ((p - 1) / tot_scnt))) % tot_scnt + 1; \
    }

void usage();
double atof();
void srand48();
long lrand48();


main(argc, argv)
     int argc;
     char **argv;
{

  double sf = 0.1;          /* scale factor      */
  long supp;               /* the i-th supplier */
  long pkey;               /* partkey           */
  long maxpkey;              /* highest partkey   */
  long ps_skey;              /* ps_suppkey        */

  if (argc < 2) {
    usage();
    exit(-1);
  }

  /* seed the random number generator */
```

```
  srand48(getpid());

  sf = atof(argv[1]);
  maxpkey = (long) (sf * PS_PER_SF);
  supp = lrand48() % 4;
  pkey = lrand48() % maxpkey + 1;

  PART_SUPP_BRIDGE(ps_skey, pkey, supp);

  fprintf(stdout, "%ld %ld", pkey, ps_skey);

  exit(0);
}


void usage()
{
  fprintf(stderr, "Usage: randpsup <SF>\n\n");
}
```

gettime.c

```
#ifdef RCSID
static char *RCSid =
   "$Header: gettime.c 15-jul-99.14:27:44 mpoess Exp $ ";
#endif /* RCSID */

/* Copyright (c) Oracle Corporation 1999. All Rights Reserved. */

/*

   NAME
     gettime.c

   DESCRIPTION
     get wall clock time.
     get cpu time.

   FUNCTIONS
     get wall clock time.
     get cpu time.

   NOTES
     Both routines return time in seconds as a double.
   MODIFIED   (MM/DD/YY)
   mpoess      07/15/99 - Creation
   mpoess      07/15/99 - Creation

*/


/*
**  Options:
**     TIME_W_TIMES:      implement gettime() with times().
**     TIME_W_GETTIME:    implement gettime() with gettimeofday().
**     CPU_W_TIMES:       implement getcpu()  with times().
**     CPU_W_GETRU:       implement getcpu()  with getrusage().
**     GETRU_STATS:       collect getrusage statistics
**     GET_P_STATS:       collect get_process_stats statistics
*/

#define SUN_OS5

#if defined(SUN_OS5)
#define TIME_W_GETTIME
#define CPU_W_TIMES
#undef GETRU_STATS
#undef CPU_W_GETRU
#endif /* SUN_OS5 */

#if defined(sequent) || defined(SEQ_PSX)
# define GET_P_STATS
#endif /* sequent */

#if defined(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)
    diffru (&ru, &kidsru);
  else
    diffru (&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)
    diffru (&ru, &kidsru);
  else
    diffru (&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_utime.tv_sec * 1000 +
                  (ru->ru_utime.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);

}


diffru (ru2, ru)

struct rusage *ru2;
struct rusage *ru;

{

  ru2->ru_utime.tv_sec -= ru->ru_utime.tv_sec;
  ru2->ru_utime.tv_usec -= ru->ru_utime.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, "%lu ", ps->ps_utime.tv_sec * 1000 +
                  (ps->ps_utime.tv_usec/1000));
  fprintf (fp, "%lu ", ps->ps_stime.tv_sec * 1000 +
                  (ps->ps_stime.tv_usec/1000));
  fprintf (fp, "%lu ", ps->ps_maxrss);
  fprintf (fp, "%lu ", ps->ps_pagein);
  fprintf (fp, "%lu ", ps->ps_reclaim);
  fprintf (fp, "%lu ", ps->ps_zerofill);
  fprintf (fp, "%lu ", ps->ps_pffincr);
  fprintf (fp, "%lu ", ps->ps_pffdecr);
  fprintf (fp, "%lu ", ps->ps_swap);
  fprintf (fp, "%lu ", ps->ps_syscall);
  fprintf (fp, "%lu ", ps->ps_volcsw);
  fprintf (fp, "%lu ", ps->ps_involcsw);
  fprintf (fp, "%lu ", ps->ps_signal);
  fprintf (fp, "%lu ", ps->ps_lread);
  fprintf (fp, "%lu ", ps->ps_lwrite);
  fprintf (fp, "%lu ", ps->ps_bread);
  fprintf (fp, "%lu ", ps->ps_bwrite);
  fprintf (fp, "%lu ", ps->ps_phread);
  fprintf (fp, "%lu",  ps->ps_phwrite);

}


diffru (ru2, ru)

struct process_stats *ru2;
struct process_stats *ru;

{

  ru2->ps_utime.tv_sec -= ru->ps_utime.tv_sec;
  ru2->ps_utime.tv_usec -= ru->ps_utime.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 */
```

end_acid.sh

```
#!/bin/ksh
#
# $Header: end_acid.sh 08-aug-99.17:06:20 mpoess Exp $
#
# end_acid.sh
#
#  Copyright (c) Oracle Corporation 1999. All Rights Reserved.
#
#    NAME
#      end_acid.sh - <one-line expansion of the name>
#
#  DESCRIPTION
#      end_cons.sh <pid of the durability run>
#      Options: See usage below
#
#    NOTES
#      <other useful comments, qualifications, etc.>
#
#    MODIFIED   (MM/DD/YY)
#    mpoess      08/08/99 - Creation
#    mpoess      08/08/99 - Creation
#

. $KIT_DIR/env

OH=$ORACLE_HOME
# ACID_DIR=$OH/tpcd/audit set in env
OUT_DIR=$ACID_OUT/
DURA_DIR=$ACID_OUT/dura
RUN_ID_FILE=$ACID_DIR/run_id

ITER=100
STEM=2
PROG=${ACID_DIR}/atranspl
IN=${ACID_DIR}/acid_in
DURA=${OUT_DIR}/dura
OUT=${OUT_DIR}/drate
DSMPL=${DURA_DIR}/durasmpl
KEY=${DURA_DIR}/key${1}_
USER=tpcd/tpcd
TRIG=1
HCNT=duracnta

# get history count

sqlplus $USER @cnt_hist > $DURA_DIR/$HCNT 2>&1

# perform the consistency

i=0
while [ $i -lt $STEM ]
do
  for j in `head -10 ${KEY}${i} | awk '{printf "%d ",$1}'`
  do
    sqlplus tpcd/tpcd @$KIT_DIR/acid/consistency/consist $j >>
$DURA_DIR/duraconsa
  done
  i=`expr $i + 1`
done

i=0
while [ $i -lt $STEM ]
do
sample.sh $DURA${i} > ${DSMPL}${i} 2>&1
i=`expr $i + 1`
done
```

run_acid.sh

```
#!/bin/ksh -x
#
# $Header: run_acid.sh 08-aug-99.15:30:10 mpoess Exp $
#
# run_acid.sh
#
#  Copyright (c) Oracle Corporation 1999. All Rights Reserved.
#
#    NAME
#      run_acid.sh - <one-line expansion of the name>
#
#  DESCRIPTION
#      Usage: run_acid.sh [-n iter] [-s stream] [-p prog] [-i infile]
#                         [-o outfile] [-d durafile] [-u usr/pswd]
#                         [-t trigger] [-f scale factor] -h
#
#      Options: See usage below
#
#    MODIFIED   (MM/DD/YY)
#    mpoess      08/08/99 - Creation
#    mpoess      08/08/99 - Creation
#

. $KIT_DIR/env

OH=$ORACLE_HOME
ACID_DIR=$ACID_DIR
OUT_DIR=$ACID_OUT

usage() {

  echo ""
  echo "Usage: $0 [-n iter] [-s stream] [-p prog] [-i infile] [-o outfile]"
  echo "          [-d durafile] [-u usr/pswd] -h"
  echo ""
  echo "-n iter     : number of iterations, default is 100"
  echo "-s stream   : number of streams, default is 2"
  echo "-p prog     : program to run, default is atranspl.ott"
  echo "-i infile   : input file prefix, suffix by process number within a"
  echo "              stream and run ID, default is ./acid_in"
  echo "-o outfile  : output file prefix, similar to input file"
  echo "              default is ./out/acid_out"
  echo "-d durafile : durability file prefix, used for durability tests"
  echo "              default is ./dura/acid_dura"
  echo "-u usr/pswd : user/password combo for database access, default
is tpcd/tpcd"
  echo "-t trigger  : trigger time between process starts, default is 1
second"
  echo "-h          : print this usage summary"
  exit 1;
}

ITER=100
STEM=2
SF=1
PROG=/oracle/kit/utils/atranspl
```

```
IN=${ACID_DIR}/acid_in
DURA_DIR=$ACID_OUT/dura
OUT=$ACID_OUT/drate
DURA=$ACID_OUT/dura
KEY=${DURA_DIR}/key$$_
USER=tpcd/tpcd
TRIG=1
HCNT=duracntb

set -- `getopt "n:s:p:i:o:d:u:ht:f:" "$@"` || usage

# get all the options

while :
do
  case "$1" in
  -n) shift; ITER=$1;;
  -s) shift; STEM=$1;;
  -p) shift; PROG=$1;;
  -i) shift; IN=$1;;
  -o) shift; OUT=$1;;
  -d) shift; DURA=$1;;
  -u) shift; USER=$1;;
  -h) usage; exit 0;;
  -t) shift; TRIG=$1;;
  -f) shift; SF=$1;;
  --) break;;
  esac
  shift;
done


echo "Starting ACID run..."

i=0
T=`expr $STEM \* $TRIG + 6`

# Get history count before the run

sqlplus $USER @cnt_hist > $DURA_DIR/$HCNT 2>&1

while [ $i -lt $STEM ]
do
  randkey 300 ${SF} u${USER} > ${KEY}${i} &
  i=`expr $i + 1`
done
wait

# perform the consistency

i=0
while [ $i -lt $STEM ]
do
  for j in `head -10 ${KEY}${i} | awk '{printf "%d ",$1}'`
  do
    sqlplus tpcd/tpcd @$KIT_DIR/acid/consistency/consist $j >>
$DURA_DIR/duraconsb
  done
  i=`expr $i + 1`
done

i=0
while [ $i -lt $STEM ]
do

  $PROG $i $STEM 1 0 i${KEY}${i} o${OUT}${i} d${DURA}${i}
u$USER s1 &
  T=`expr $T - $TRIG`
  i=`expr $i + 1`

done
wait

echo "ACID run completed"
```

sample.sh

```
#!/bin/ksh
#
# $Header: sample.sh 08-aug-99.17:10:00 mpoess Exp $
#
# sample.sh
#
#  Copyright (c) Oracle Corporation 1999. All Rights Reserved.
#
#   NAME
#     sample.sh - <one-line expansion of the name>
#
#   DESCRIPTION
#     <short description of component this file declares/defines>
#
#   NOTES
#     <other useful comments, qualifications, etc.>
#
#   MODIFIED   (MM/DD/YY)
#   mpoess      08/08/99 - Creation
#   mpoess      08/08/99 - Creation
#

# $1 durability output file

. $KIT_DIR/env

cat $1 | grep o_key | awk '{printf "%d \n", $2}' > /tmp/okey$$
cat $1 | grep l_key | awk '{printf "%d \n", $2}' > /tmp/lkey$$

paste /tmp/okey$$ /tmp/lkey$$ > /tmp/keys$$
tail -6 /tmp/keys$$ > /tmp/6keys$$

echo "Keys chosen are:"
cat /tmp/6keys$$

i=1
while [ $i -le 6 ]
do

j=`cat /tmp/6keys$$ | tail -${i} | head -1`
sqlplus tpcd/tpcd @sample $j
i=`expr $i + 1`
done

/bin/rm -f /tmp/*key*
```

sample.sql

```
Rem
Rem $Header: sample.sql 08-aug-99.17:10:34 mpoess Exp $
Rem
Rem sample.sql
Rem
Rem  Copyright (c) Oracle Corporation 1999. All Rights Reserved.
Rem
Rem    NAME
Rem      sample.sql - <one-line expansion of the name>
Rem
Rem    DESCRIPTION
Rem      <short description of component this file declares/defines>
Rem
```

```
Rem    NOTES
Rem      <other useful comments, qualifications, etc.>
Rem
Rem    MODIFIED   (MM/DD/YY)
Rem    mpoess      08/08/99 - Creation
Rem    mpoess      08/08/99 - Created
Rem
alter session set nls_date_format = 'YYYY-MM-DD HH:MI:SS';
select * from history where  h_o_key = &&1 and h_l_key = &&2;

exit;
```

Appendix D: Query Text and Query Output

1.log

```
Begin Execution at Tue Sep 24 18:19:36 2002


-- using default substitutions

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
AVG_PRICE            AVG_DISC             COUNT_ORDER
A            F           37734107.00          56586554400.73
53758257134.87       55909065222.83       25.52
38273.13             0.05                 1478493.00
N            F           991417.00            1487504710.38
1413082168.05        1469649223.19        25.52
38284.47             0.05                 38854.00
N            O           74476040.00          111701729697.74
106118230307.61      110367043872.50      25.50
38249.12             0.05                 2920374.00
R            F           37719753.00          56568041380.90
53741292684.60       55889619119.83       25.51
38250.85             0.05                 1478870.00


4 rows processed.
Statement Processed in 0.05 seconds.

Ended Executing this Query at Tue Sep 24 18:19:36 2002


Query Started at 1032909576.96
Query Ended at 1032909577.01
Query Processed in 0.05 seconds


SQL statements processed: 1
Queries processed: 1
```

2.log

```
Begin Execution at Tue Sep 24 18:19:37 2002


-- using default substitutions

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 ps_supplycost = (
select
min(ps_supplycost)
from
partsupp,
supplier,
nation,
region
where
p_partkey = ps_partkey
and s_suppkey = ps_suppkey
and s_nationkey = n_nationkey
and n_regionkey = r_regionkey
and r_name = 'EUROPE'
)
order by
s_acctbal desc,
n_name,
s_name,
p_partkey
)
where rownum <= 100

S_ACCTBAL            S_NAME               N_NAME
P_PARTKEY            P_MFGR
S_ADDRESS                         S_PHONE
S_COMMENT
9938.53              Supplier#000005359       UNITED KINGDOM
185358.00            Manufacturer#4
QKuHYh,vZGiwu2FWEJoLDx04               33-429-790-6131
blithely silent pinto beans are furiously. slyly final deposits acros
9937.84              Supplier#000005969       ROMANIA
108438.00            Manufacturer#1
ANDENSOSmk,miq23Xfb5RWt6dvUcvt6Qa       29-520-692-3537
carefully slow deposits use furiously. slyly ironic platelets above the
ironic
9936.22              Supplier#000005250       UNITED KINGDOM
249.00               Manufacturer#4
B3rqp0xbSEim4Mpy2RH J                  33-320-228-2957
blithely special packages are. stealthily express deposits across the
closely final instructi
```

9923.77 Supplier#000002324 GERMANY
29821.00 Manufacturer#4
y3OD9UywSTOk 17-779-299-1839
quickly express packages breach quiet pinto beans. requ
9871.22 Supplier#000006373 GERMANY
43868.00 Manufacturer#5
J8fcXWsTqM 17-813-485-8637
never silent deposits integrate furiously blit
9870.78 Supplier#000001286 GERMANY
81285.00 Manufacturer#2
YKA,E2fjiVd7eUrzp2Ef8j1QxGo2DFnosaTEH 17-516-924-4574
final theodolites cajole slyly special,
9870.78 Supplier#000001286 GERMANY
181285.00 Manufacturer#4
YKA,E2fjiVd7eUrzp2Ef8j1QxGo2DFnosaTEH 17-516-924-4574
final theodolites cajole slyly special,
9852.52 Supplier#000008973 RUSSIA
18972.00 Manufacturer#2
t5L67YdBYYH6o,Vz24jpDyQ9 32-188-594-7038
quickly regular instructions wake-- carefully unusual braids into the
expres
9847.83 Supplier#000008097 RUSSIA
130557.00 Manufacturer#2
xMe97bpE69NzdwLoX 32-375-640-3593
slyly regular dependencies sleep slyly furiously express dep
9847.57 Supplier#000006345 FRANCE
86344.00 Manufacturer#1
VSt3rzk3qG698u6ld8HhOByvrTcSTSvQlDQDag 16-886-766-7945
silent pinto beans should have to snooze carefully along the final reques
9847.57 Supplier#000006345 FRANCE
173827.00 Manufacturer#2
VSt3rzk3qG698u6ld8HhOByvrTcSTSvQlDQDag 16-886-766-7945
silent pinto beans should have to snooze carefully along the final reques

--- Additional lines deleted per Comment in Clause 8.3.3.4.

100 rows processed.
Statement Processed in 0.53 seconds.

Ended Executing this Query at Tue Sep 24 18:19:37 2002

Query Started at 1032909577.09
Query Ended at 1032909577.61
Query Processed in 0.53 seconds

SQL statements processed: 1
Queries processed: 1

3.log

Begin Execution at Tue Sep 24 18:50:18 2002

-- using default substitutions

```
select * from (
select
l_orderkey,
sum(l_extendedprice * (1 - l_discount)) as revenue,
o_orderdate,
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 O_ORDERDATE
O_SHIPPRIORITY
2456423.00 406181.01 1995-03-05 0.00
3459808.00 405838.70 1995-03-04 0.00
492164.00 390324.06 1995-02-19 0.00
1188320.00 384537.94 1995-03-09 0.00
2435712.00 378673.06 1995-02-26 0.00
4878020.00 378376.80 1995-03-12 0.00
5521732.00 375153.92 1995-03-13 0.00
2628192.00 373133.31 1995-02-22 0.00
993600.00 371407.46 1995-03-05 0.00
2300070.00 367371.15 1995-03-13 0.00

10 rows processed.
Statement Processed in 1.67 seconds.

Ended Executing this Query at Tue Sep 24 18:50:19 2002

Query Started at 1032911418.05
Query Ended at 1032911419.72
Query Processed in 1.67 seconds

SQL statements processed: 1
Queries processed: 1

4.log

Begin Execution at Tue Sep 24 18:50:21 2002

-- using default substitutions

```
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 10594.00
2-HIGH 10476.00

```
3-MEDIUM        10410.00
4-NOT SPECIFIED 10556.00
5-LOW           10487.00


5 rows processed.
Statement Processed in 1.46 seconds.

Ended Executing this Query at Tue Sep 24 18:50:23 2002


Query Started at 1032911421.71
Query Ended at 1032911423.18
Query Processed in 1.46 seconds


SQL statements processed: 1
Queries processed: 1
```

5.log

```
Begin Execution at Tue Sep 24 18:19:37 2002


-- using default substitutions

select
n_name,
sum(l_extendedprice * (1 - l_discount)) as revenue
from
customer,
orders,
lineitem,
supplier,
nation,
region
where
c_custkey = o_custkey
and l_orderkey = o_orderkey
and l_suppkey = s_suppkey
and c_nationkey = s_nationkey
and s_nationkey = n_nationkey
and n_regionkey = r_regionkey
and r_name = 'ASIA'
and o_orderdate >= 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
INDONESIA                  55502041.17
VIETNAM                    55295087.00
CHINA                      53724494.26
INDIA                      52035512.00
JAPAN                      45410175.70


5 rows processed.
Statement Processed in 0.26 seconds.

Ended Executing this Query at Tue Sep 24 18:19:37 2002


Query Started at 1032909577.68
Query Ended at 1032909577.94
Query Processed in 0.26 seconds


SQL statements processed: 1
Queries processed: 1
```

6.log

```
Begin Execution at Tue Sep 24 18:19:38 2002


-- using default substitutions

select
sum(l_extendedprice * 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
123141078.23


1 row processed.
Statement Processed in 0.01 seconds.

Ended Executing this Query at Tue Sep 24 18:19:38 2002


Query Started at 1032909578.01
Query Ended at 1032909578.02
Query Processed in 0.01 seconds


SQL statements processed: 1
Queries processed: 1
```

7.log

```
Begin Execution at Tue Sep 24 18:19:38 2002


-- using default substitutions

select
supp_nation,
cust_nation,
l_year,
sum(volume) as revenue
from
(
select
n1.n_name as supp_nation,
n2.n_name as cust_nation,
                                        to_number (to_char
(l_shipdate,'yyyy')) as l_year,
l_extendedprice * (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,
l_year
order by
supp_nation,
cust_nation,
l_year

SUPP_NATION          CUST_NATION          L_YEAR
REVENUE
FRANCE               GERMANY              1995.00
54639732.73
FRANCE               GERMANY              1996.00
54633083.31
GERMANY              FRANCE               1995.00
52531746.67
GERMANY              FRANCE               1996.00
52520549.02


4 rows processed.
Statement Processed in 0.37 seconds.

Ended Executing this Query at Tue Sep 24 18:19:38 2002


Query Started at 1032909578.09
Query Ended at 1032909578.46
Query Processed in 0.37 seconds


SQL statements processed: 1
Queries processed: 1
```

8.log

```
Begin Execution at Tue Sep 24 18:19:38 2002


-- using default substitutions

select
o_year,
sum(case when nation='BRAZIL' then volume else 0 end )/
sum(volume)
as mkt_share
from
(
select
to_number (to_char (o_orderdate, 'yyyy')) as o_year,
l_extendedprice * (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
o_year
order by
o_year

O_YEAR               MKT_SHARE
1995.00              0.03
1996.00              0.04


2 rows processed.
Statement Processed in 1.80 seconds.

Ended Executing this Query at Tue Sep 24 18:19:40 2002


Query Started at 1032909578.52
Query Ended at 1032909580.32
Query Processed in 1.80 seconds


SQL statements processed: 1
Queries processed: 1
```

9.log

```
Begin Execution at Tue Sep 24 18:19:40 2002


-- using default substitutions

select
nation,
o_year,
sum(amount) as sum_profit
from
(
select
n_name as nation,
to_number (to_char (o_orderdate, 'yyyy')) as o_year,
l_extendedprice * (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,
o_year
order by
nation,
o_year desc

NATION                O_YEAR              SUM_PROFIT
ALGERIA               1998.00             31342867.23
ALGERIA               1997.00             57138193.02
ALGERIA               1996.00             56140140.13
ALGERIA               1995.00             53051469.65
ALGERIA               1994.00             53867582.13
ALGERIA               1993.00             54942718.13
ALGERIA               1992.00             54628034.71
ARGENTINA             1998.00             30211185.71
ARGENTINA             1997.00             50805741.75
ARGENTINA             1996.00             51923746.58
--- Additional lines deleted per Comment in Clause 8.3.3.4.


175 rows processed.
Statement Processed in 7.79 seconds.

Ended Executing this Query at Tue Sep 24 18:19:48 2002


Query Started at 1032909580.39
Query Ended at 1032909588.18
Query Processed in 7.79 seconds


SQL statements processed: 1
Queries processed: 1
```

10.log

```
Begin Execution at Tue Sep 24 18:19:48 2002


-- using default substitutions

select  *  from  (
select
c_custkey,
c_name,
sum(l_extendedprice * (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
57040.00            Customer#000057040       734235.25
632.87              JAPAN
Eioyzjf4pp                      22-895-641-3466
requests sleep blithely about the furiously i
143347.00           Customer#000143347       721002.69
2557.47             EGYPT
1aReFYv,Kw4                     14-742-935-3718
fluffily bold excuses haggle finally after the u
60838.00            Customer#000060838       679127.31
2454.77             BRAZIL
64EaJ5vMAHWJlBOxJklpNc2RJiWE    12-913-494-9813
furiously even pinto beans integrate under the ruthless foxes; ironic,
even dolphins across the slyl
101998.00           Customer#000101998       637029.57
3790.89             UNITED KINGDOM
01c9CILnNtfOQYmZj               33-593-865-6378
accounts doze blithely! enticing, final deposits sleep blithely special
accounts. slyly express accounts pla
125341.00           Customer#000125341       633508.09
4983.51             GERMANY
S29ODD6bceU8QSuuEJznkNaK        17-582-695-5962
quickly express requests wake quickly blithely
25501.00            Customer#000025501       620269.78
7725.04             ETHIOPIA
 W556MXuoiaYCCZamJI,Rn0B4ACUGdkQ8DZ     15-874-808-6793
quickly special requests sleep evenly among the special deposits. special
deposi
115831.00           Customer#000115831       596423.87
5098.10             FRANCE
rFeBbEEyk dl ne7zV5fDrmiq1oK09wV7pxqCgIc 16-715-386-3788
carefully bold excuses sleep alongside of the thinly idle
84223.00            Customer#000084223       594998.02
528.65              UNITED KINGDOM
nAVZCs6BaWap rrM27N 2qBnzc5WBauxbA       33-442-824-8191
pending, final ideas haggle final requests. unusual, regular asymptotes
affix according to the even foxes.
54289.00            Customer#000054289       585603.39
5583.02             IRAN
vXCxoCsU0Bad5JQI ,oobkZ                  20-834-292-4707
express requests sublate blithely regular requests. regular, even ideas
solve.
39922.00            Customer#000039922       584878.11
7321.11             GERMANY
Zgy4s50l2GKN4pLDPBU8m342gIw6R            17-147-757-8036
even pinto beans haggle. slyly bold accounts inte
6226.00             Customer#000006226       576783.76
2230.09             UNITED KINGDOM
8gPu8,NPGkfyQQ0hcIYUGPIBWc,ybP5g,        33-657-701-3391
quickly final requests against the regular instructions wake blithely final
instructions. pa
922.00              Customer#000000922       576767.53
3869.25             GERMANY
Az9RFaut7NkPnc5zSD2PwHgVwr4jRzq          17-945-916-9648
boldly final requests cajole blith
147946.00           Customer#000147946       576455.13
2030.13             ALGERIA
iANyZHjqhyy7Ajah0pTrYyhJ                 10-886-956-3143
furiously even accounts are blithely above the furiousl
115640.00           Customer#000115640       569341.19
6436.10             ARGENTINA
```

```
Vtgfia9qI 7EpHgecU1X                    11-411-543-4901
final instructions are slyly according to the
73606.00            Customer#000073606      568656.86
1785.67             JAPAN
xuR0Tro5yChDfOCrjkd2ol                  22-437-653-6966
furiously bold orbits about the furiously busy requests wake across the
furiously quiet theodolites. d
110246.00           Customer#000110246      566842.98
7763.35             VIETNAM
7KzflgX MDOq7sOkI                       31-943-426-9837
dolphins sleep blithely among the slyly final
142549.00           Customer#000142549      563537.24
5085.99             INDONESIA
ChqEoK43OysjdHbtKCp6dKqjNyvvi9          19-955-562-2398
regular, unusual dependencies boost slyly; ironic attainments nag
fluffily into the unusual packages?
146149.00           Customer#000146149      557254.99
1791.55             ROMANIA
s87fvzFQpU                              29-744-164-6487
silent, unusual requests detect quickly slyly regul
52528.00            Customer#000052528      556397.35
551.79              ARGENTINA
NFztyTOR10UOJ                           11-208-192-3205
unusual requests detect. slyly dogged theodolites use slyly. deposit
23431.00            Customer#000023431      554269.54
3381.86             ROMANIA
HgiV0phqhaIa9aydNoIlb                   29-915-458-2654
instructions nag quickly. furiously bold accounts cajol


20 rows processed.
Statement Processed in 3.06 seconds.

Ended Executing this Query at Tue Sep 24 18:19:51 2002


Query Started at 1032909588.33
Query Ended at 1032909591.39
Query Processed in 3.06 seconds


SQL statements processed: 1
Queries processed: 1
```

11.log

```
Begin Execution at Tue Sep 24 18:19:51 2002


-- using default substitutions

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) * 0.0001000000
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
129760.00           17538456.86
166726.00           16503353.92
191287.00           16474801.97
161758.00           16101755.54
34452.00            15983844.72
139035.00           15907078.34
9403.00             15451755.62
154358.00           15212937.88
38823.00            15064802.86
--- Additional lines deleted per Comment in Clause 8.3.3.4.


1048 rows processed.
Statement Processed in 3.33 seconds.

Ended Executing this Query at Tue Sep 24 18:19:54 2002


Query Started at 1032909591.51
Query Ended at 1032909594.84
Query Processed in 3.33 seconds


SQL statements processed: 1
Queries processed: 1
```

12.log

```
Begin Execution at Tue Sep 24 18:19:54 2002


-- using default substitutions

select
        l_shipmode,
        sum(case
                when o_orderpriority = '1-URGENT'
                        or o_orderpriority = '2-HIGH'
                        then 1
                else 0
        end) as high_line_count,
        sum(case
                when o_orderpriority <> '1-URGENT'
                        and o_orderpriority <> '2-HIGH'
                        then 1
                else 0
        end) as low_line_count
from
        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       6202.00               9324.00
SHIP       6200.00               9262.00


2 rows processed.
Statement Processed in 0.02 seconds.

Ended Executing this Query at Tue Sep 24 18:19:54 2002


Query Started at 1032909594.90
Query Ended at 1032909594.92
Query Processed in 0.02 seconds


SQL statements processed: 1
Queries processed: 1
```

13.log

```
Begin Execution at Tue Sep 24 18:19:54 2002


-- using default substitutions

select
c_count,
count(*) as custdist
from
(
select
c_custkey,
count(o_orderkey)  as  c_count
from
customer, orders where
c_custkey = o_custkey(+)
and o_comment(+) not like '%special%requests%'
group by
c_custkey
) c_orders
group by
c_count
order by
custdist desc,
c_count desc

C_COUNT             CUSTDIST
0.00                50004.00
9.00                6641.00
10.00               6566.00
11.00               6058.00
8.00                5949.00
12.00               5553.00
13.00               4989.00
19.00               4748.00
7.00                4707.00
18.00               4625.00
15.00               4552.00
17.00               4530.00
14.00               4484.00
20.00               4461.00
16.00               4323.00
21.00               4217.00
22.00               3730.00
6.00                3334.00
23.00               3129.00
24.00               2622.00
25.00               2079.00
5.00                1972.00
26.00               1593.00
27.00               1185.00
4.00                1033.00
28.00               869.00
29.00               559.00
3.00                398.00
30.00               373.00
31.00               235.00
2.00                144.00
32.00               128.00
33.00               71.00
34.00               48.00
35.00               33.00
1.00                23.00
36.00               17.00
37.00               7.00
40.00               4.00
38.00               4.00
39.00               2.00
41.00               1.00


42 rows processed.
Statement Processed in 7.17 seconds.

Ended Executing this Query at Tue Sep 24 18:20:02 2002


Query Started at 1032909595.00
Query Ended at 1032909602.17
Query Processed in 7.17 seconds


SQL statements processed: 1
Queries processed: 1
```

14.log

```
Begin Execution at Tue Sep 24 18:20:02 2002


-- using default substitutions

select
        100.00 * sum(case
                when p_type like 'PROMO%'
                        then l_extendedprice * (1 - l_discount)
                else 0
        end) / sum(l_extendedprice * (1 - l_discount)) as
promo_revenue
from
        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
16.38


1 row processed.
Statement Processed in 0.02 seconds.

Ended Executing this Query at Tue Sep 24 18:20:02 2002


Query Started at 1032909602.24
```

Query Ended at 1032909602.27
Query Processed in 0.02 seconds

SQL statements processed: 1
Queries processed: 1

15.log

Begin Execution at Tue Sep 24 18:20:02 2002

-- using default substitutions

```
create view revenue0 (supplier_no, total_revenue) as
select
l_suppkey,
sum(l_extendedprice * (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 0.02 seconds.

```
select
s_suppkey,
s_name,
s_address,
s_phone,
total_revenue
from
supplier,
revenue0
where
s_suppkey = supplier_no
and total_revenue = (
select
max(total_revenue)
from
revenue0
)
order by
s_suppkey
```

S_SUPPKEY S_NAME
S_ADDRESS S_PHONE TOTAL_REVENUE
8449.00 Supplier#000008449
Wp34zim9qYFbVctdW 20-469-856-8873 1772627.21

1 row processed.
Statement Processed in 2.21 seconds.

```
drop view revenue0
```

Statement Processed in 0.03 seconds.

Ended Executing this Query at Tue Sep 24 18:20:04 2002

Query Started at 1032909602.34
Query Ended at 1032909604.60
Query Processed in 2.26 seconds

SQL statements processed: 3
Queries processed: 1

16.log

Begin Execution at Tue Sep 24 18:20:04 2002

-- using default substitutions

```
select
p_brand,
p_type,
p_size,
count(distinct ps_suppkey) 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_suppkey not in (
select
s_suppkey
from
supplier
where
s_comment like '%Customer%Complaints%'
)
group by
p_brand,
p_type,
p_size
order by
supplier_cnt desc,
p_brand,
p_type,
p_size
```

P_BRAND P_TYPE P_SIZE SUPPLIER_CNT
Brand#41 MEDIUM BRUSHED TIN 3.00 28.00
Brand#54 STANDARD BRUSHED COPPER 14.00 27.00
Brand#11 STANDARD BRUSHED TIN 23.00 24.00
Brand#11 STANDARD BURNISHED BRASS 36.00 24.00
Brand#15 MEDIUM ANODIZED NICKEL 3.00 24.00
Brand#15 SMALL ANODIZED BRASS 45.00 24.00
Brand#15 SMALL BURNISHED NICKEL 19.00 24.00
Brand#21 MEDIUM ANODIZED COPPER 3.00 24.00
Brand#22 SMALL BRUSHED NICKEL 3.00 24.00
Brand#22 SMALL BURNISHED BRASS 19.00 24.00
--- Additional lines deleted per Comment in Clause 8.3.3.4.

18314 rows processed.
Statement Processed in 7.13 seconds.

Ended Executing this Query at Tue Sep 24 18:20:11 2002

Query Started at 1032909604.67
Query Ended at 1032909611.80
Query Processed in 7.13 seconds

SQL statements processed: 1
Queries processed: 1

17.log

Begin Execution at Tue Sep 24 18:20:11 2002

-- using default substitutions

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

AVG_YEARLY
348406.05


1 row processed.
Statement Processed in 1.42 seconds.

Ended Executing this Query at Tue Sep 24 18:20:13 2002


Query Started at 1032909611.88
Query Ended at 1032909613.30
Query Processed in 1.42 seconds


SQL statements processed: 1
Queries processed: 1
```

18.log

```
Begin Execution at Tue Sep 24 18:20:13 2002


-- using default substitutions

select  *  from  (
select
c_name,
c_custkey,
o_orderkey,
o_orderdate,
o_totalprice,
sum(l_quantity)
from
customer,
orders,
lineitem
where
o_orderkey in (
select
l_orderkey
from
lineitem
group by
l_orderkey having
sum(l_quantity) > 300
)
and c_custkey = o_custkey
and o_orderkey = l_orderkey
group by
c_name,
c_custkey,
o_orderkey,
o_orderdate,
o_totalprice
order by
o_totalprice desc,
o_orderdate
)
where rownum <= 100

C_NAME                C_CUSTKEY           O_ORDERKEY
O_ORDERDATE
O_TOTALPRICE          SUM(L_QUANTITY)
Customer#000128120       128120.00           4722021.00
1994-04-07
544089.09             323.00
Customer#000144617       144617.00           3043270.00
1997-02-12
530604.44             317.00
Customer#000013940       13940.00            2232932.00          1997-
04-13
522720.61             304.00
Customer#000066790       66790.00            2199712.00          1996-
09-30
515531.82             327.00
Customer#000046435       46435.00            4745607.00          1997-
07-03
508047.99             309.00
Customer#000015272       15272.00            3883783.00          1993-
07-28
500241.33             302.00
Customer#000146608       146608.00           3342468.00
1994-06-12
499794.58             303.00
Customer#000096103       96103.00            5984582.00          1992-
03-16
494398.79             312.00
Customer#000024341       24341.00            1474818.00          1992-
11-15
491348.26             302.00
Customer#000137446       137446.00           5489475.00
1997-05-23
487763.25             311.00
Customer#000107590       107590.00           4267751.00
1994-11-04
485141.38             301.00
Customer#000050008       50008.00            2366755.00          1996-
12-09
483891.26             302.00
--- Additional lines deleted per Comment in Clause 8.3.3.4.


57 rows processed.
Statement Processed in 1.76 seconds.

Ended Executing this Query at Tue Sep 24 18:20:15 2002


Query Started at 1032909613.36
Query Ended at 1032909615.12
Query Processed in 1.76 seconds


SQL statements processed: 1
Queries processed: 1
```

19.log

```
Begin Execution at Tue Sep 24 18:20:15 2002
```

```
-- using default substitutions

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

REVENUE
3083843.06


1 row processed.
Statement Processed in 1.39 seconds.

Ended Executing this Query at Tue Sep 24 18:20:16 2002


Query Started at 1032909615.18
Query Ended at 1032909616.57
Query Processed in 1.39 seconds


SQL statements processed: 1
Queries processed: 1
```

20.log

```
Begin Execution at Tue Sep 24 18:20:16 2002


-- using default substitutions

select
s_name,
s_address
from
supplier,
nation
where
s_suppkey in (
select
ps_suppkey
from
partsupp
where
ps_partkey in (
select
p_partkey
from
parts
where
p_name like 'forest%'
)
and ps_availqty > (
select
0.5 * sum(l_quantity)
from
lineitem
where
l_partkey = ps_partkey
and l_suppkey = ps_suppkey
and l_shipdate >= to_date ('1994-01-01', 'YYYY-MM-DD')
and l_shipdate < add_months( to_date ('1994-01-01', 'YYYY-MM-
DD'), 12)
)
)
and s_nationkey = n_nationkey
and n_name = 'CANADA'
order by
s_name

S_NAME                 S_ADDRESS
Supplier#000000020       iybAE,RmTymrZVYaFZva2SH,j
Supplier#000000091
YV45D7TkfdQanOOZ7q9QxkyGUapU1oOWU6q3
Supplier#000000197       YC2Acon6kjY3zj3Fbxs2k4Vdf7X0cd2F
Supplier#000000226       83qOdU2EYRdPQAQhEtn GRZEd
Supplier#000000285       Br7e1nnt1yxrw6ImgpJ7YdhFDjuBf
Supplier#000000378       FfbhyCxWvcPrO8ltp9
Supplier#000000402       i9Sw4DoyMhzhKXCH9By,AYSgmD
Supplier#000000530       0qwCMwobKY OcmLyfRXlagA8ukENJv,
Supplier#000000688       D fw5ocppmZpYBBIPI718hCihLDZ5KhKX
Supplier#000000710       f19YPvOyb QoYwjKC,oPycpGfieBAcwKJo
Supplier#000000736
l6i2nMwVuovfKnuVgaSGK2rDy65DlAFLegiL7
--- Additional lines deleted per Comment in Clause 8.3.3.4.


204 rows processed.
Statement Processed in 2.77 seconds.

Ended Executing this Query at Tue Sep 24 18:20:19 2002


Query Started at 1032909616.70
Query Ended at 1032909619.47
Query Processed in 2.77 seconds


SQL statements processed: 1
Queries processed: 1
```

21.log

```
Begin Execution at Tue Sep 24 18:20:19 2002


-- using default substitutions

select * from (
```

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

S_NAME                NUMWAIT
Supplier#000002829       20.00
Supplier#000005808       18.00
Supplier#000000262       17.00
Supplier#000000496       17.00
Supplier#000002160       17.00
Supplier#000002301       17.00
Supplier#000002540       17.00
Supplier#000003063       17.00
Supplier#000005178       17.00
Supplier#000008331       17.00
--- Additional lines deleted per Comment in Clause 8.3.3.4.


100 rows processed.
Statement Processed in 39.67 seconds.

Ended Executing this Query at Tue Sep 24 18:20:59 2002


Query Started at 1032909619.55
Query Ended at 1032909659.21
Query Processed in 39.67 seconds


SQL statements processed: 1
Queries processed: 1
```

22.log

```
Begin Execution at Tue Sep 24 18:20:59 2002


-- using default substitutions

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

CNTRYCODE NUMCUST              TOTACCTBAL
13        888.00               6737713.99
17        861.00               6460573.72
18        964.00               7236687.40
23        892.00               6701457.95
29        948.00               7158866.63
30        909.00               6808436.13
31        922.00               6806670.18


7 rows processed.
Statement Processed in 2.97 seconds.

Ended Executing this Query at Tue Sep 24 18:21:02 2002


Query Started at 1032909659.30
Query Ended at 1032909662.27
Query Processed in 2.97 seconds


SQL statements processed: 1
Queries processed: 1
```

Appendix E: Seed and Input Parameters

NOTE: Some of the scripts are very long, and, in order to save space, are printed in very small font. The electronic version of the FDR has the complete listings. But all of the scripts may not be readable in the paper version.

Stream00 Seed

930050338

Stream00

```
14      1996-10-01
2       16      BRASS   AMERICA
9       saddle
20      pale    1994-01-01      ROMANIA
6       1994-01-01      0.05    25
17      Brand#43  MED DRUM
18      314
8       VIETNAM         ASIA    ECONOMY POLISHED
COPPER
21      FRANCE
13      special   accounts
3       BUILDING        1995-03-08
22      22      31      18      25      13      32
        21
16      Brand#34  LARGE PLATED    50      8       13
        16      12      27      11      37
4       1994-03-01
11      IRAN    0.0000010000
15      1997-03-01
1       79
10      1994-09-01
19      Brand#34  Brand#24  Brand#33  7       18      30
5       AMERICA         1994-01-01
7       INDONESIA       VIETNAM
12      REG AIR   FOB   1997-01-01
```

Stream01 Seed

930050339

Stream01

```
21      UNITED KINGDOM
3       HOUSEHOLD       1995-03-24
18      315
5       ASIA    1994-01-01
11      UNITED KINGDOM  0.0000010000
7       ARGENTINA       JORDAN
6       1994-01-01      0.02    25
20      black   1997-01-01      INDIA
17      Brand#45  JUMBO BAG
12      FOB     REG AIR   1996-01-01
16      Brand#14  PROMO POLISHED  21      30      48
        2       6       19      12      25
15      1994-12-01
13      special   deposits
10      1993-06-01
2       3       NICKEL  EUROPE
8       JORDAN  MIDDLE EAST     ECONOMY BURNISHED
COPPER
14      1997-01-01
19      Brand#41  Brand#52  Brand#22  3       19      26
9       puff
22      18      33      19      15      12      24
        31
1       87
4       1996-10-01
```

Stream02 Seed

930050340

Stream02

```
6       1994-01-01      0.07    24
17      Brand#42  JUMBO PKG
14      1997-05-01
16      Brand#54  MEDIUM ANODIZED      37      16
        32      40      14      42      38      36
19      Brand#43  Brand#45  Brand#22  8       20      22
10      1994-03-01
9       papaya
2       41      TIN     AMERICA
15      1997-06-01
8       ETHIOPIA        AFRICA  LARGE BRUSHED
COPPER
5       EUROPE  1994-01-01
22      22      16      21      27      24      12
        11
12      MAIL    AIR     1997-01-01
7       CHINA   ETHIOPIA
13      special   deposits
18      313
1       95
4       1994-07-01
20      lime    1996-01-01      UNITED KINGDOM
3       AUTOMOBILE      1995-03-10
11      IRAQ    0.0000010000
21      MOROCCO
```

Stream03 Seed

930050341

Stream03

```
8       RUSSIA  EUROPE  LARGE POLISHED TIN
5       MIDDLE EAST     1994-01-01
4       1997-02-01
6       1994-01-01      0.05    25
17      Brand#44  JUMBO DRUM
7       IRAN    RUSSIA
1       103
18      314
22      14      12      15      17      18      10
        20
14      1997-08-01
9       navajo
10      1994-12-01
15      1995-03-01
11      UNITED STATES   0.0000010000
```

20 steel 1994-01-01 JORDAN 10 1994-07-01
2 29 COPPER MIDDLE EAST 9 lemon
21 INDIA
19 Brand#45 Brand#23 Brand#21 3 10 29
13 special deposits
16 Brand#34 ECONOMY BURNISHED 42 46
17 6 11 27 34 15
12 TRUCK AIR 1997-01-01
3 FURNITURE 1995-03-26

Stream04 Seed

930050342

Stream04

5 AFRICA 1994-01-01
21 ALGERIA
14 1997-11-01
19 Brand#52 Brand#11 Brand#15 9 11 25
15 1997-10-01
17 Brand#41 WRAP BAG
12 RAIL AIR 1997-01-01
6 1994-01-01 0.02 25
4 1994-10-01
9 medium
8 KENYA AFRICA LARGE BURNISHED TIN
16 Brand#14 STANDARD POLISHED 3 40
44 31 7 10 41 19
11 JAPAN 0.0000010000
2 17 STEEL AMERICA
10 1993-09-01
18 312
1 111
13 special deposits
7 BRAZIL KENYA
22 18 13 16 25 17 21
11
3 AUTOMOBILE 1995-03-12
20 frosted 1993-01-01 BRAZIL

Stream05 Seed

930050343

Stream05

21 PERU
15 1995-06-01
4 1997-05-01
6 1995-01-01 0.08 24
7 ROMANIA FRANCE
16 Brand#54 MEDIUM BRUSHED 49 38
18 12 36 3 7 10
19 Brand#54 Brand#44 Brand#14 4 12 21
18 313
14 1993-02-01
22 10 16 28 18 15 34
21
11 ALGERIA 0.0000010000
13 pending packages
3 FURNITURE 1995-03-28
1 119
2 5 NICKEL MIDDLE EAST
5 AMERICA 1995-01-01
8 FRANCE EUROPE MEDIUM BRUSHED TIN
20 purple 1996-01-01 PERU
12 AIR REG AIR 1995-01-01
17 Brand#43 WRAP PKG

Appendix F: Benchmark Scripts

dbtables.sql

```
set numwidth 25
SELECT COUNT(*) FROM LINEITEM;

SELECT * FROM LINEITEM
WHERE L_ORDERKEY IN
 ( 4, 26598,  148577,  387431, 56704, 517442, 600000)
AND L_LINENUMBER = 1
ORDER BY L_ORDERKEY;



SELECT * FROM REGION;

SELECT COUNT(*) FROM NATION;

SELECT * FROM NATION
WHERE N_NATIONKEY IN (3,10,14,20)
ORDER BY N_NATIONKEY;



SELECT COUNT(*) FROM ORDERS;

SELECT * FROM ORDERS
WHERE O_ORDERKEY IN ( 7, 44065, 287590, 411111, 483876,
599942 )
ORDER BY O_ORDERKEY;



SELECT COUNT(*) FROM PARTS;

SELECT * FROM PARTS
WHERE P_PARTKEY IN (1,984,8743,9028,13876,17899,20000)
ORDER BY P_PARTKEY;



SELECT COUNT(*) FROM PARTSUPP;

SELECT* FROM PARTSUPP
  WHERE PS_PARTKEY = 3398
  AND PS_SUPPKEY = (SELECT MIN(PS_SUPPKEY)
    FROM PARTSUPP WHERE PS_PARTKEY = 3398);

SELECT* FROM PARTSUPP
  WHERE PS_PARTKEY =15873
  AND PS_SUPPKEY = (SELECT MIN(PS_SUPPKEY)
    FROM PARTSUPP WHERE PS_PARTKEY = 15873);

SELECT* FROM PARTSUPP
  WHERE PS_PARTKEY = 11394
  AND PS_SUPPKEY = (SELECT MIN(PS_SUPPKEY)
    FROM PARTSUPP WHERE PS_PARTKEY = 11394);

SELECT* FROM PARTSUPP
  WHERE PS_PARTKEY = 6743
  AND PS_SUPPKEY = (SELECT MIN(PS_SUPPKEY)
    FROM PARTSUPP WHERE PS_PARTKEY = 6743);

SELECT* FROM PARTSUPP
  WHERE PS_PARTKEY = 19763
  AND PS_SUPPKEY = (SELECT MIN(PS_SUPPKEY)
    FROM PARTSUPP WHERE PS_PARTKEY =19763);

SELECT COUNT(*) FROM SUPPLIER;

SELECT * FROM SUPPLIER
WHERE  S_SUPPKEY IN (83,265,492,784,901,1000)
ORDER BY S_SUPPKEY;



DROP TABLE MINMAX;

CREATE TABLE MINMAX
(TNAME CHAR(15),
KEYMIN INTEGER,
KEYMAX INTEGER);

INSERT INTO MINMAX
SELECT
'LINEITEM_ORD',MIN(L_ORDERKEY),MAX(L_ORDERKEY)
FROM LINEITEM ;

INSERT INTO MINMAX
SELECT
'LINEITEM_NBR',MIN(L_LINENUMBER),MAX(L_LINENUMBER)
FROM LINEITEM;

INSERT INTO MINMAX
SELECT 'ORDERTBL',MIN(O_ORDERKEY),MAX(O_ORDERKEY)
FROM ORDERS;

INSERT INTO MINMAX
SELECT 'CUSTOMER',MIN(C_CUSTKEY),MAX(C_CUSTKEY)
FROM CUSTOMER;

INSERT INTO MINMAX
SELECT 'PART',MIN(P_PARTKEY),MAX(P_PARTKEY)
FROM PARTS;

INSERT INTO MINMAX
SELECT 'SUPPLIER',MIN(S_SUPPKEY),MAX(S_SUPPKEY)
FROM SUPPLIER;

INSERT INTO MINMAX
SELECT
'PARTSUPP_PART',MIN(PS_PARTKEY),MAX(PS_PARTKEY)
FROM PARTSUPP;

INSERT INTO MINMAX
SELECT
'PARTSUPP_SUPP',MIN(PS_SUPPKEY),MAX(PS_SUPPKEY)
FROM PARTSUPP ;

INSERT INTO MINMAX
SELECT 'NATION',MIN(N_NATIONKEY),MAX(N_NATIONKEY)
FROM NATION;

INSERT INTO MINMAX
SELECT 'REGION',MIN(R_REGIONKEY),MAX(R_REGIONKEY)
FROM REGION;

SELECT * FROM MINMAX;

exit;
```

gen_seed.sh

```
LOG_FILE=$1
SDATE=$2
SEED_FILE=$3
echo "Setting the random number seed at $SDATE" >> ${LOG_FILE}
```

```
PSEED=`date +%m:%d:%H:%M:%S | sed -e 's/://g'`
echo "Using ${PSEED} as seed0" >> ${LOG_FILE}
echo ${PSEED} >  $SEED_FILE
echo "Done setting the random number seed at `date`" >>
${LOG_FILE}
```

gtime.c

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

# include <sys/time.h>

main ()
{

struct timeval tp ;

  gettimeofday(&tp, NULL) ;

  printf ("%ld\n", tp.tv_sec) ;
/*
  printf ("usec = %ld\n", tp.tv_usec) ;
*/
}

/*
double gettime();
main() {
  printf("%f", gettime());
  exit(0);
}
*/
```

qexecpl.c

```
#ifdef RCSID
static char *RCSid =
   "$Header: qexecpl.c 05-aug-99.13:54:50 mpoess Exp $ ";
#endif /* RCSID */

/* Copyright (c) Oracle Corporation 1999. All Rights Reserved. */

/*

   NAME
     qexecpl.c - <one-line expansion of the name>

   DESCRIPTION
     SQL Execution Engine, Oracle v8,  OCI version

   PRIVATE FUNCTION(S)
     <list of static functions defined in .c file - with one-line
descriptions>

   MODIFIED   (MM/DD/YY)
     mpoess    08/05/99 - make compile
     mpoess    11/13/98 - fix pddl statement
     pswong    02/19/97 - migrating to version 8
     pswong    04/02/96 - more polishing
     pswong    03/25/96 - polish up
     pswong    03/06/96 - 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 "qexecpl.h"

/* Function Prototypes */

extern double gettime();

/* 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 OFEN();
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.         */
#ifdef LINUX
```

```
char cmnt1[1024];           /* Buffer to save the comment.        */
int first=1;
#endif

#ifdef LINUX
FILE *qtemp;      /* fd for query template              */
FILE *logfile;   /* log and report files               */
FILE *rep;
#else
FILE *qtemp = stdin;      /* fd for query template              */
FILE *logfile = stdout;   /* log and report files               */
FILE *rep = stdout;
#endif
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 */

OCIEnv *tpcenv = NULL;
OCIServer *tpcsrv = NULL;
OCIError *errhp = NULL;
OCISvcCtx *tpcsvc = NULL;
OCISession *tpcusr = NULL;
OCIStmt *curq = NULL;
OCIStmt *cur_dml = NULL;
OCIStmt *cur_ddl = NULL;
OCIParam *tpcpar = NULL;

sword status = OCI_SUCCESS;  /* OCI return value */

/* usage: prints the usage of the program */

void usage() {

  fprintf(stderr,"\nUsage: qexec username/password [q<path name for
query template file>]\n");
  fprintf(stderr,"               [l<path name for log>] [r<path name for
reports>]\n\n");
  fprintf(stderr,"Options:\n");
  fprintf(stderr,"q<path for query>         : full path name for the query
template file.\n");
  fprintf(stderr,"                          (default is stdin)\n");
  fprintf(stderr,"l<path name for log>      : full path name for log
files\n");
  fprintf(stderr,"                          (default is stdout)\n");
  fprintf(stderr,"r<path name for reports>  : full path name for
reports\n");
  fprintf(stderr,"                          (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)
    OCIError *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) OCIErrorGet(errhp,1,NULL,(sb4*)&errcode,(text*)msg,
                                          2048,OCI_HTYPE_ERROR);
    else
      (void) OCIErrorGet(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) OCIErrorGet(errhp,1,NULL,(sb4*)&errcode,(text*)msg,
                                          2048,OCI_HTYPE_ERROR);
    else
      (void) OCIErrorGet(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) OCIErrorGet(errhp,1,NULL,(sb4*)&errcode,(text*)msg,
                                          2048,OCI_HTYPE_ERROR);
    else
      (void) OCIErrorGet(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);
}

#ifdef LINUX
int main(argc,argv)
#else
void main(argc,argv)
#endif
    int argc;
    char *argv[];
{

  int i;
  int retcode;      /* Return code for get_statement */

#ifdef LINUX
  logfile=fopen("/dev/stdout","w");
  qtemp=fopen("/dev/stdin","rw");
  rep=fopen("/dev/stdout","w");
#endif

  /* 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\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) {
#ifdef LINUX
       first=0;
#endif
              end_flag = 0;      /* reset query end flag */
              /* save the comment so that we can print it out later on  */
              strcpy(cmnt, stmt);
              break;
     }
#ifdef LINUX
      if(first==0)
      {
       sprintf(cmnt1, "%s%s", cmnt, stmt);
              strcpy(cmnt, cmnt1);
      }
      else
      {
#endif
       fprintf(logfile, "%s", stmt);
       fprintf(rep, "%s", stmt);
#ifdef LINUX
      }
#endif
      break;

      /* if this is a set_row_fetch command */
    case SET_FETCHROW:
      fprintf(logfile,"Setting the number of rows to fetch to: %ld\n\n",
                num_to_fetch);
      break;

      /* if this is a SQL statement */
    case SQL_STMT:

      /* Executes the query */

      SQLexec();

      s_tr_end = gettime();
      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 = gettime();

  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;

  /* 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] = (dltype *) memalloc (sizeof(dltype));
      dlist[i]->defhdl = NULL;
/*      OCIhalloc(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,(dvoid *)0,0,0,0);

  if((status=OCIEnvInit((OCIEnv **)&tpcenv,OCI_DEFAULT,0,(dvoid
**)0)) !=
     OCI_SUCCESS)
    sql_error(tpcenv, status, 0);

  OCIhalloc(tpcenv,&errhp,OCI_HTYPE_ERROR);
  OCIhalloc(tpcenv,&curq,OCI_HTYPE_STMT);
  OCIhalloc(tpcenv,&cur_dml,OCI_HTYPE_STMT);
  OCIhalloc(tpcenv,&cur_ddl,OCI_HTYPE_STMT);
  OCIhalloc(tpcenv,&tpcsvc,OCI_HTYPE_SVCCTX);
  OCIhalloc(tpcenv,&tpcsrv,OCI_HTYPE_SERVER);
  OCIhalloc(tpcenv,&tpcusr,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(tpcusr,OCI_HTYPE_SESSION,logname,strlen(logname),OCI_
ATTR_USERNAME,
                errhp);

OCIaset(tpcusr,OCI_HTYPE_SESSION,passwd,strlen(passwd),OCI_A
TTR_PASSWORD,
                errhp);

  if ((status = OCISessionBegin(tpcsvc, errhp, tpcusr,
OCI_CRED_RDBMS,
                                        OCI_DEFAULT)) !=
OCI_SUCCESS)
    sql_error(errhp,status,1);


OCIaset(tpcsvc,OCI_HTYPE_SVCCTX,tpcusr,0,OCI_ATTR_SESSIO
N,errhp);

  /* Enable session parallel dml */

  sprintf((char *) stmt, PDMLTXT);

  OCIStmtPrepare(cur_dml,errhp,(text *)stmt,strlen((char *)stmt),
          OCI_NTV_SYNTAX,OCI_DEFAULT);
  OCIsexec(tpcsvc,cur_dml,errhp,1);

  /* Enable session parallel ddl */

  sprintf((char *) stmt, PDDLTXT);

  OCIStmtPrepare(cur_ddl,errhp,(text *)stmt,strlen((char *)stmt),
          OCI_NTV_SYNTAX,OCI_DEFAULT);
  OCIsexec(tpcsvc,cur_ddl,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\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 = gettime();

    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 = gettime();

  /* prepare the statement */

  if ((status = OCIStmtPrepare(curq, errhp, (text*) stmt, (ub4)
strlen(stmt),
                                        OCI_NTV_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 thie version        */


OCIaget(curq,OCI_HTYPE_STMT,&stmttyp,NULL,OCI_ATTR_STM
T_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);

  OCIsexec(tpcsvc,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 statments will screw it up */


OCIaget(curq,OCI_HTYPE_STMT,&rows_ret,NULL,OCI_ATTR_RO
W_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(tpcsvc,errhp);
  OCIhfree(tpcenv,OCI_HTYPE_STMT);
  OCIhfree(tpcsvc,OCI_HTYPE_SVCCTX);
  OCIhfree(tpcsrv,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].dbsize),
                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] = (dltype *) memalloc(sizeof(dltype));
      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) {
              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,1);
  }
  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_RO
W_COUNT,errhp);

    print_rows(num,rows_ret);

    /* To avoid 1022 from OFEN */
    /* 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;
      }
    }
  } else {
    OCIStmtFetch(curq, errhp, ntf, OCI_FETCH_NEXT,
OCI_DEFAULT);

OCIaget(curq,OCI_HTYPE_STMT,&rows_ret,NULL,OCI_ATTR_RO
W_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, qtemp) != 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 = ' ';
}
```

qexecpl.h

```
/*
 * $Header: qexecpl.h 05-aug-99.13:54:55 mpoess Exp $
 */

/* Copyright (c) Oracle Corporation 1999. All Rights Reserved. */

/* NOTE:  See 'header_template.doc' in the 'doc' dve under the 'forms'
      directory for the header file template that includes instructions.
*/

/*
   NAME
     qexecpl.h

   DESCRIPTION
     SQL statement execution front-end header file.

   PUBLIC FUNCTION(S)
     <list of external functions declared/defined - with one-line
descriptions>

   PRIVATE FUNCTION(S)
     <list of static functions defined in .c file - with one-line
descriptions>

   EXAMPLES

   NOTES
     <other useful comments, qualifications, etc.>

   MODIFIED   (MM/DD/YY)
   mpoess      08/05/99 - make compile
   mpoess      07/15/99 - Creation
   mpoess      07/15/99 - Creation
*/

/*
# ifndef S_ORACLE
#  include <s.h>
# endif
*/
#ifndef 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>

#ifndef OCIDFN
#include <ocidfn.h>
#endif /* OCIDFN */

#ifndef 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

#ifndef TRUE
#define TRUE 1
#endif /* TRUE */

#ifndef FALSE
#define FALSE 1
#endif /* FALSE */

#ifndef LINUX
#define MAX(x,y) ((x >= y) ? x : y)
#define MIN(x,y) ((x <= y) ? x : y)
#endif

/* 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; */
  OCITypeCode dbtype;
/*  text buf[MAX_COLNAME_SIZE]; */
  text *buf;
  ub1 precision;
} sltype;


/* defines and typedefs for query select list definition */

#define MAX_ARRAY 50       /* 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

#ifndef NULL
#define NULL 0
#endif

#ifndef NULLP
# define NULLP (void *)NULL
#endif /* NULLP */

#ifndef DISCARD
# define DISCARD (void)
#endif

#ifndef sword
# define sword int
#endif

#ifndef 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 OCIaget(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 OCIaset(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,errh,iter,0,NULL,NULL,OCI_DE
FAULT)) != 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
160)"
#define PDDLTXT "alter session force parallel ddl parallel (degree 64)"

#endif /* QSTREAMPL_H */
```

runTPCRall.sh

Note: The script runTPCRall.sh as disclosed performs the load test and two performance runs. In order to facilitate rebooting the system after the load test and after the first performance run, the script was run 3 different times. The first time it was executed, both performance runs were commented out so that only the load test was executed. The second
time it was executed, the load test and the second performance run were commented out so that only the first performance run was executed. The third time it was executed, the load test and the first performance run were commented out so that only the second performance run was executed.

```
#!/bin/ksh

ECHO=echo

sqlplus=$ORACLE_HOME/bin/sqlplus
GTIME=${KIT_DIR}/utils/gtime
DATABASE_USER=tpcd/tpcd
SCALE_FACTOR=$1

RUN_ID_FILE=${KIT_DIR}/audit/r_id

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

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

OUT_DIR=${KIT_DIR}/audit/tests/${RUN_ID}
if [ ! -d $OUT_DIR ]
then
  mkdir $OUT_DIR
fi

SCRIPT_LOG_FILE=${OUT_DIR}/main.out
RDB_TABLES=${OUT_DIR}/rdbtablest.log
FIRST_TEN=${OUT_DIR}/firstten.log
CHECK_IDX=${OUT_DIR}/checkidx.log

echo Start TPC-R Benchmark SEQUENCE NUMBER: $RUN_ID >
$SCRIPT_LOG_FILE
echo >> $SCRIPT_LOG_FILE
echo "Starting a new Oracle log file: alert_tpcr100g.log" >>
$SCRIPT_LOG_FILE
echo >> $SCRIPT_LOG_FILE
mv $ORACLE_HOME/rdbms/log/alert_tpcr100g.log
$ORACLE_HOME/rdbms/log/alert_tpcr100g.log.preAudit.$RUN_ID
touch $ORACLE_HOME/rdbms/log/alert_tpcr100g.log

echo "Start: load database `date`" >> $SCRIPT_LOG_FILE

load-notime.sh > ${OUT_DIR}/create_database_notimed.log

tstart.sh

STIME=`$GTIME`
echo "Start: timed load portion `date`" >> $SCRIPT_LOG_FILE

load-time.sh > ${OUT_DIR}/create_database_timed.log

echo "End: timed load portion `date`" >> $SCRIPT_LOG_FILE

$KIT_DIR/audit/gen_seed.sh $KIT_DIR/audit/seed-log "`date`"
$KIT_DIR/audit/seed

echo Generated seed: `cat $KIT_DIR/audit/seed` >>
$SCRIPT_LOG_FILE

echo "Start: dbtables.sql, firstten.sql, checkidx.sql (BEFORE) `date`"
>> $SCRIPT_LOG_FILE
$sqlplus ${DATABASE_USER} @$KIT_DIR/audit/dbtables >
${RDB_TABLES}.before_run 2>&1
$sqlplus ${DATABASE_USER} @$KIT_DIR/audit/firstten >
${FIRST_TEN}.before_run 2>&1
$sqlplus ${DATABASE_USER} @$KIT_DIR/audit/checkidx >
${CHECK_IDX}.before_run 2>&1
echo "End: dbtables.sql, firstten.sql, checkidx.sql (BEFORE) `date`" >>
$SCRIPT_LOG_FILE

runTPCRpt ${SCALE_FACTOR} 1 ${RUN_ID}

runTPCRpt ${SCALE_FACTOR} 2 ${RUN_ID}

echo "Start: dbtables.sql and checkidx.sql (AFTER) `date`" >>
$SCRIPT_LOG_FILE
$sqlplus ${DATABASE_USER} @$KIT_DIR/audit/dbtables >
${RDB_TABLES}.after_run 2>&1
$sqlplus ${DATABASE_USER} @$KIT_DIR/audit/checkidx >
${CHECK_IDX}.after_run 2>&1
echo "End: dbtables.sql and checkidx.sql (AFTER) `date`" >>
$SCRIPT_LOG_FILE

kill.sh tpcr100g

cp -a $ORACLE_HOME/rdbms/log/alert_tpcr100g.log $OUT_DIR

echo "End TPC-R Benchmark SEQUENCE NUMBER: $RUN_ID
`date`" >> $SCRIPT_LOG_FILE
```

runTPCRpt

```
#!/bin/ksh
#ECHO=/bin/echo
SCRIPT_DIR=${KIT_DIR}/scripts
SQL_DIR=${KIT_DIR}/sql
UPD_DIR=${KIT_DIR}/update
SRC_DIR=${KIT_DIR}/utils
QRY_DIR=${KIT_DIR}/queries # this is the location of the query
template file
QGEN_DIR=${KIT_DIR}/dbgen
QGEN=${QGEN_DIR}/qgen
QEXEC=${SRC_DIR}

DSS_QUERY=${KIT_DIR}/queries
export DSS_QUERY

UPD_SQL=${UPD_DIR}/sql
UPD_SPT=${UPD_DIR}/scripts
UPD_SRC=${UPD_DIR}/source
UPD_DAT=${UPD_DIR}/data

TPCR_BIN=${KIT_DIR}/audit/bin

GTIME=${SRC_DIR}/gtime
SEED_FILE=${KIT_DIR}/audit/seed

DF=/dev/null
HID=1
INTERVAL=60
COUNT=1200

# The defaults

QPROG=${QEXEC}/qexec

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 "          <scale factor> <run_number>"
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 $U2PROG."
echo "-o               : Collect Oracle statistics."
echo "-s               : Collect System statistics."
echo "-u <user/passwd> : User/Password. Default is tpch/tpch."
echo "-h               : Displays this message."
}
set -- `getopt "p:u1:u2:osu:h" "$@"` || 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 "3" ]
then
  usage
  exit 1
fi

SF=$1
PARA=$2
RUN_ID=$3
NUM_STREAMS=5

OUT_DIR=${KIT_DIR}/audit/tests/${RUN_ID}
#OUT_DIR=/flatfiles/results/${RUN_ID}
if [ ! -d $OUT_DIR ]
then
        mkdir $OUT_DIR
fi

TPCR_LOG=${OUT_DIR}
TPCR_RPT=${OUT_DIR}
OUT=${OUT_DIR}

let UF_SET="($PARA-1)*($NUM_STREAMS+1)+1"
START_SET=1
let STOP_SET=$NUM_STREAMS
let START_SET_UPDATE="($PARA-1)*($NUM_STREAMS+1)+2"
let
STOP_SET_UPDATE="$START_SET_UPDATE+$NUM_STREAMS
-1"

TPCR_LOG_FILE=${TPCR_LOG}/m${PARA}s0
TPCR_RPT_FILE=${TPCR_RPT}/m${PARA}s0inter
QRY_FILE=${TPCR_RPT}/qtemp.${PARA}s0
QUERY_PARAMETER=${TPCR_LOG}/qp${PARA}.0
SCRIPT_LOG_FILE=${TPCR_LOG}/m${PARA}timing
UF1_LOG=${TPCR_LOG}/m${PARA}s0rf1
UF2_LOG=${TPCR_LOG}/m${PARA}s0rf2
STREAM_COUNT_LOG=${TPCR_LOG}/m${PARA}tstrcnt

echo "TPC-R Test - RUN:${PARA} SEQUENCE:${RUN_ID} `date`"
> $SCRIPT_LOG_FILE
echo "TPC-R Test - RUN:${PARA} SEQUENCE:${RUN_ID} `date`"
> $TPCR_RPT_FILE
echo "Generates query template file with seed: `cat $SEED_FILE` for
stream 0" >> $SCRIPT_LOG_FILE
echo >> $SCRIPT_LOG_FILE

${QGEN} -c -r `cat $SEED_FILE` -p 0 -s ${SF} -l
$QUERY_PARAMETER > ${QRY_FILE}
cat ${QRY_FILE} | grep -v "Rem" > /tmp/.tmp_qfile
cp /tmp/.tmp_qfile ${QRY_FILE}
rm -f /tmp/.tmp_qfile

# Bouncing the database before the power test

tstart.sh >>  $SCRIPT_LOG_FILE

START=`$GTIME`
echo "Start Power Test - RUN:${PARA} SEQUENCE:${RUN_ID}
Execution Starts $START, `date`" >>  $SCRIPT_LOG_FILE
echo "" >> $SCRIPT_LOG_FILE

# Execute UF1

SDATE=`date`
UF1_START=`$GTIME`
echo "Start UF1  $UF1_START, `date`" >> $SCRIPT_LOG_FILE

${ECHO} ${UPD_SPT}/runuf1.sh ${UF_SET} >> $UF1_LOG 2>&1
# Execute Query Stream

UF1_END=`${GTIME}`
E1DATE=`date`

UF1_TIME=`echo $UF1_END - $UF1_START | bc`
echo UF1: Execution Time: $UF1_TIME >> ${TPCR_RPT_FILE}
echo Start Time: $UF1_START, $SDATE >> ${TPCR_RPT_FILE}
echo End Time: $UF1_END, $E1DATE >> ${TPCR_RPT_FILE}
echo "" >> ${TPCR_RPT_FILE}

echo "End UF1 $UF1_END, ${E1DATE}" >> $SCRIPT_LOG_FILE
echo >> $SCRIPT_LOG_FILE

echo "Start Query Part `$GTIME`, `date` " >> $SCRIPT_LOG_FILE

${QPROG} ${DATABASE_USER} q${QRY_FILE}
l${TPCR_LOG_FILE} r${TPCR_RPT_FILE}  > $DF  2>&1

# Execute UF2

UF2_START=`${GTIME}`
E2DATE=`date`

echo "End Query Part `$GTIME`, ${E2DATE}" >>
$SCRIPT_LOG_FILE
echo "" >> $SCRIPT_LOG_FILE

echo "Start UF2  $UF1_START, `date`" >> $SCRIPT_LOG_FILE
${ECHO} ${UPD_SPT}/runuf2.sh ${UF_SET} >> $UF2_LOG 2>&1
UF2_END=`${GTIME}`
END=`${GTIME}`
EDATE=`date`
```

```
echo "End UF2 $UF2_END, $EDATE" >>  $SCRIPT_LOG_FILE
echo >>  $SCRIPT_LOG_FILE

echo "End TPC-R Power Test - RUN:${PARA}
SEQUENCE:${RUN_ID}, $END, $EDATE" >> $SCRIPT_LOG_FILE
MEA_INT=`echo $END - $START | bc`
echo "Elapsed Time for TPC-R Power Test - RUN:${PARA}
SEQUENCE:${RUN_ID} is $MEA_INT" >> $SCRIPT_LOG_FILE
echo >> $SCRIPT_LOG_FILE

UF2_TIME=`echo $UF2_END - $UF2_START | bc`
echo UF2: Execution Time: $UF2_TIME >> ${TPCR_RPT_FILE}
echo Start Time: $UF2_START, ${E2DATE} >> ${TPCR_RPT_FILE}
echo End Time: $UF2_END, $EDATE >> ${TPCR_RPT_FILE}

${KIT_DIR}/audit/abridge.pl ${TPCR_LOG_FILE}

i=$START_SET
PSEED=`cat $SEED_FILE`

while [ $i -le $STOP_SET ]; do
  TPCR_LOG_FILE=${TPCR_LOG}/mt${RUN_ID}_${i}.log
  TPCR_RPT_FILE=${TPCR_RPT}/mt${RUN_ID}_${i}.rpt
            QUERY_PARAMETER=${TPCR_LOG}/qp${PARA}.${i}
  QRY_FILE=${TPCR_RPT}/qtemp.${PARA}s${i}

  PSEED=`expr $PSEED + 1`
  ${QGEN} -c -r ${PSEED} -p ${i} -s ${SF} -l
$QUERY_PARAMETER > ${QRY_FILE}
  cat ${QRY_FILE} | grep -v "Rem" > /tmp/.tmp_qfile
  cp /tmp/.tmp_qfile ${QRY_FILE}
  rm -f /tmp/.tmp_qfile
  i=`expr $i + 1`
done

# Bouncing the database before the throughtput test

tstart.sh >>  $SCRIPT_LOG_FILE

TH_START_D=`date`
TH_START_T=`${GTIME}`
echo >> $SCRIPT_LOG_FILE

rm -f /tmp/th_pipe1
mknod /tmp/th_pipe1 p
rm -f /tmp/th_pipe2
mknod /tmp/th_pipe2 p
i=$START_SET

echo "Start Throughput Test - RUN:${PARA}
SEQUENCE:${RUN_ID} $TH_START_T, $TH_START_D"  >>
$SCRIPT_LOG_FILE

# starts a script to count the streams during the throughput run
(scnt.sh $PARA $RUN_ID > $STREAM_COUNT_LOG &)

while [ $i -le $STOP_SET ]; do
  M_SDATE=`date`
  M_STIME=`${GTIME}`
            TPCR_LOG_FILE=${TPCR_LOG}/m${PARA}s${i}
            TPCR_RPT_FILE=${TPCR_RPT}/m${PARA}s${i}inter
  echo "Start Query Stream $i $M_STIME, ${M_SDATE}" >>
$SCRIPT_LOG_FILE
            QRY_FILE=${TPCR_RPT}/qtemp.${PARA}s${i}
  ${QPROG} ${DATABASE_USER} q${QRY_FILE}
l${TPCR_LOG_FILE} r${TPCR_RPT_FILE} | grep -v "Connected to
ORACLE" >> $SCRIPT_LOG_FILE &
  i=`expr $i + 1`
done

(${KIT_DIR}/audit/runTPCRus $RUN_ID $START_SET_UPDATE
$STOP_SET_UPDATE ${SF} $PARA >> $SCRIPT_LOG_FILE 2>&1
&)

wait
THQ_END_T=`$GTIME`
THQ_END_D=`date`
echo End all Query Streams $THQ_END_T, $THQ_END_D >>
$SCRIPT_LOG_FILE
print > /tmp/th_pipe1
read < /tmp/th_pipe2

TH_END_D=`date`
TH_END_T=`$GTIME`
echo End Update Stream ${TH_END_T}, ${TH_END_D} >>
$SCRIPT_LOG_FILE
echo >> $SCRIPT_LOG_FILE
echo "End Throughput Test ${TH_END_T}, ${TH_END_D}" >>
$SCRIPT_LOG_FILE
echo Execution Time Throughput Test: `echo ${TH_END_T} -
${TH_START_T} | bc` >> $SCRIPT_LOG_FILE

i=$START_SET
while [ $i -le $STOP_SET ]; do
            TPCR_LOG_FILE=${TPCR_LOG}/m${PARA}s${i}
  ${KIT_DIR}/audit/abridge.pl ${TPCR_LOG_FILE}
            i=`expr $i + 1`
done
kill -9 `ps -ef | grep scnt.sh | grep -v grep | awk '{print $2}'`
#${SF}/scripts/kill.sh scnt.sh
#calculate the metric
#analyze_streams.pl -f p -n $RUN_ID >
${TPCR_RPT}/tpch_metric.${RUN_ID}.${HID}.rpt
```

runuf1.sh

```
#!/bin/ksh
#
# $Header: runuf1.sh 25-oct-2001.15:56:04 mpoess Exp $
#
# runuf1.sh
#
# Copyright (c) 1999, 2001, Oracle Corporation.  All rights reserved.
#
#    NAME
#      runuf1.sh - <one-line expansion of the name>
#
# DESCRIPTION
#    runuf1.sh -l [<path name for reports>] -u [<uid/passwd>]
#              -p [<program>] <run_id> <scale factor> <pair number>
#              <parallelism>
# USAGE
#    To execute UF1.
#
#    NOTES
#      <other useful comments, qualifications, etc.>
#
#    MODIFIED   (MM/DD/YY)
#    mpoess      10/25/01 - change default directory for update sets
#    mpoess      10/17/01 - add support for external tables
#    mpoess      08/15/99 - Creation
#    mpoess      08/15/99 - Creation
#
#
. $KIT_DIR/env
O=${ORACLE_HOME}
UPDATE_DIR=${KIT_DIR}/update
SCRIPT_DIR=${UPDATE_DIR}/scripts
LOG_DIR=${UPDATE_DIR}/log
GTIME=${SCRIPT_DIR}/gtime
SF=${SCALE_FACTOR}
PAR_HINT=${UPDATE_DOP}

LOGPATH=.
PASSWD=${DATABASE_USER}
```

```
if [ $# -lt 1 ];
then
  echo runuf1.sh setnum
  exit 1
fi
SETNUM=$1
i=1
PID=""

# perform the update function 1

START=`$GTIME`

# first create the temp tables

sqlplus /NOLOG << !

connect $PASSWD;
set timing  on
set serveroutput on
set echo on

drop directory data_dir;
create directory data_dir as '${KIT_DIR}/update/data';

drop table temp_l_et;
create table temp_l_et(
   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          char(10) ,
   l_linenumber        number ,
   l_shipinstruct      char(25) ,
   l_comment           varchar(44)
)
organization external (
type ORACLE_LOADER
default directory data_dir
access parameters
(
            records delimited by newline
            badfile 'l_et.${SETNUM}.bad'
            logfile 'l_et.${SETNUM}.log'
            fields terminated by '|'
            missing field values are null
      )
      location (
      'lineitem.tbl.u${SETNUM}'))
reject limit unlimited;

drop table temp_o_et;
create table temp_o_et(
   o_orderdate         date ,
   o_orderkey          number ,
   o_custkey           number ,
   o_orderpriority     char(15) ,
   o_shippriority      number ,
   o_clerk             char(15) ,
   o_orderstatus       char(1) ,
   o_totalprice        number ,
   o_comment           varchar(79)
)
organization external (
type ORACLE_LOADER
default directory data_dir
access parameters
(
            records delimited by newline
            badfile 'o_et.${SETNUM}.bad'
            logfile 'o_et.${SETNUM}.log'
            fields terminated by '|'
            missing field values are null
      )
      location (
      'orders.tbl.u${SETNUM}'))
reject limit unlimited;

alter table temp_l_et parallel  ${PAR_HINT};
alter table temp_o_et parallel  ${PAR_HINT};

rem alter session set events '10132 trace name context forever, level 1';


alter session force parallel dml parallel ${PAR_HINT};
alter session set isolation_level = serializable;
alter session set optimizer_index_cost_adj = 10;

commit;

insert into orders (select * from temp_o_et);

insert into lineitem (select * from temp_l_et);

commit;

drop table temp_l_et;
drop table temp_o_et;

exit;
!

END=`$GTIME`

# Done

echo ""
echo "Update Function 1 Set $SETNUM done!"
echo "Elapsed Time is `echo $END - $START | bc`"
echo ""
```

runuf2.sh

```
#!/bin/ksh
#
# $Header: runuf2.sh 25-oct-2001.15:56:05 mpoess Exp $
#
# runuf2.sh
#
# Copyright (c) 1999, 2001, Oracle Corporation.  All rights reserved.
#
#    NAME
#      runuf2.sh - <one-line expansion of the name>
#
# DESCRIPTION
#    runuf2.sh [-u <uid/passwd to login>] [-p <program>] <run_id>
#              <scale factor> <pair number> <parallelism>
# USAGE
#    To execute UF2.
#
#    NOTES
```

```
#     <other useful comments, qualifications, etc.>
#
#   MODIFIED   (MM/DD/YY)
#   mpoess      10/25/01 - change default directory for update sets
#   mpoess      10/17/01 - add support for external tables
#   mpoess      08/15/99 - Creation
#   mpoess      08/15/99 - Creation
#
. $KIT_DIR/env
UPDATE_DIR=${KIT_DIR}/update
SCRIPT_DIR=${UPDATE_DIR}/scripts
GTIME=${SCRIPT_DIR}/gtime
LOG_DIR=${UPDATE_DIR}/log
PAR_HINT=${UPDATE_DOP}
SF=${SCALE_FACTOR}
PASSWD=${DATABASE_USER}

if [ $# -lt 1 ]
then
  usage
  exit 1
fi

SETNUM=$1

i=1
PID=""

START=`$GTIME`
# first create the temp tables

sqlplus /NOLOG << !

connect $PASSWD;
set timing  on
set serveroutput on
set echo on

drop directory data_dir;
create directory data_dir as '${KIT_DIR}/update/data';

drop table temp_okey_et;
drop table temp_okey;

create table temp_okey_et(
   t_orderkey          number
)
organization external (
type ORACLE_LOADER
default directory data_dir
access parameters
(
            records delimited by newline
            badfile 'okey.${SETNUM}.bad'
            logfile 'okey.${SETNUM}.log'
            fields terminated by '|'
            missing field values are null
      )
      location (
      'delete.${SETNUM}'))
reject limit unlimited;

alter table temp_okey_et parallel ${PAR_HINT};

create table temp_okey parallel  ${PAR_HINT} nologging as select *
from temp_okey_et;

create unique index i_temp_okey on temp_okey (t_orderkey) parallel
${PAR_HINT} nologging compute statistics;
analyze table temp_okey estimate statistics sample 2 percent;

rem alter session set events '10132 trace name context forever, level 1';
alter session  force parallel dml parallel ${PAR_HINT};
alter session set isolation_level=serializable;
alter session set optimizer_index_cost_adj=10;


delete from (select /*+ use_nl(o) */ o.rowid from orders o, temp_okey t
where o.o_orderkey = t.t_orderkey order by 1);

delete from (select /*+ use_nl(l) */ l.rowid from lineitem l,temp_okey t
where l.l_orderkey = t.t_orderkey order by 1);

commit;

drop table temp_okey;
drop table temp_okey_et;
exit;
!

END=`$GTIME`

# Done

echo ""
echo "Update Function 2 Set $SETNUM done!"
echo "Elapsed Time is `echo $END - $START | bc`"
echo ""
```

tstart.sh

```
#!/bin/ksh

DIR=`pwd`
cd $ORACLE_HOME/dbs

if [ "$1" != "" ]; then
  PFILE="pfile=$1.ora"
else
  if [ -f "$ORACLE_HOME/dbs/init$ORACLE_SID.ora" ]; then
    PFILE="pfile=$ORACLE_HOME/dbs/init$ORACLE_SID.ora"
  else
    PFILE="pfile=$INITORA"
  fi
fi


sqlplus /NOLOG << !
connect / as sysdba
shutdown immediate
startup $PFILE
exit
!

cd $DIR
```

tshut.sh

```
#!/bin/ksh

sqlplus /NOLOG << !
connect / as sysdba
shutdown immediate
exit
!
```

abridge.pl

```
#!/usr/bin/perl

$maxrows=40;

$infile=shift(@ARGV);
```

```
$infile=~m/(.*)\/(.*)$/;
$filename=$2;
$pathname=$1;
#print "$filename\n";
#print "$pathname\n";

$filename=~m/m(.*)s(.*)$/;
$run=$1;
$stream=$2;
#print "r $run\n";
#print "st $stream\n";
open(IN,$infile);
$bq=0;

#/private/tpcd/mpoess/kit/audit/tests/15/m1s0inter

$filebase=$pathname."/m$run"."s$stream"."q";
while ($l=<IN>) {
 $_ = $l;
  if (m/^-- TPC\-H\/TPC\-R(.*)\(Q(\d+)\)/) {
                        $q=$2;
   $bq=1;
   #print "found the beginning of query $q\n";
   close OUT;
                        $fname=$filebase.$q;
                        open (OUT,"> $fname");
                        push(@fnames,$fname);
                        $nsp=0;
 }
 if ($bq) {
   print OUT $l;
 }
            if ($l =~ /Statement Processed/) {
  $nsp++;
            }
 if ( ($l =~ /Statement Processed/) && (


                        ( ($q != 15) && ($nsp==1))

                        ||

                        ( ($q == 15) && ($nsp==3))



                       )){
   #print "found end of output $nr\n";
   $bq=0;
 }
}
#@fnames=split(/ /,`ls $filebase* | xargs`);
while (@fnames) {
 $file=shift(@fnames);
 #print "$file\n";
 $nl=`wc -l $file`;
            if ($nl > 200) {
                        #print ("$file\n");
  system("mv $file /tmp/qo");
                        `head -150 /tmp/qo > $file`;
                        `echo ..... delete lines ..... >> $file`;
                        `tail -50 /tmp/qo >> $file`;
 }
}
```

abridge22.pl

```
#!/usr/bin/perl

$maxrows=40;

$infile=shift(@ARGV);

$infile=~m/(.*)\/(.*)$/;
$filename=$2;
$pathname=$1;
#print "$filename\n";
#print "$pathname\n";

$filename=~m/m(.*)s(.*)$/;
$run=$1;
$stream=$2;
#print "r $run\n";
#print "st $stream\n";
open(IN,$infile);
$bq=0;

#/private/tpcd/mpoess/kit/audit/tests/15/m1s0inter

$filebase=$pathname."/m$run"."s$stream"."q";
while ($l=<IN>) {
 $_ = $l;
  if (m/^-- TPC\-D(.*)\(Q(\d+)\)/) {
                        $q=$2;
   $bq=1;
   #print "found the beginning of query $q\n";
   close OUT;
                        $fname=$filebase.$q;
                        open (OUT,"> $fname");
                        push(@fnames,$fname);
                        $nsp=0;
 }
 if ($bq) {
   print OUT $l;
 }
            if ($l =~ /Statement Processed/) {
  $nsp++;
            }
 if ( ($l =~ /Statement Processed/) && (


                        ( ($q != 15) && ($nsp==1))

                        ||

                        ( ($q == 15) && ($nsp==3))



                       )){
   #print "found end of output $nr\n";
   $bq=0;
 }
}
#@fnames=split(/ /,`ls $filebase* | xargs`);
while (@fnames) {
 $file=shift(@fnames);
 #print "$file\n";
 $nl=`wc -l $file`;
            if ($nl > 200) {
                        #print ("$file\n");
  system("mv $file /tmp/qo");
                        `head -150 /tmp/qo > $file`;
                        `echo ..... delete lines ..... >> $file`;
                        `tail -50 /tmp/qo >> $file`;
 }
}
```

Appendix H: Price Quotes

From: MaryBeth Pierantoni [mary.beth.pierantoni@oracle.com]
Sent: Tuesday, October 01, 2002 2:15 PM
To: Kong_Yang@exchange.dell.com
Cc: mary.beth.pierantoni@oracle.com; herve.lejeune@oracle.com
Subject: TPC-R Pricing for 3 years
Hi Kong:

To follow is the Oracle price for 3 years:

Product	**Price**	**Quantity**	**Extended Price**
Oracle9i Database Enterprise Edition Release 2, Named User Plus for 3 years	$10,000	4	$40,000
Partitioning, Named User Plus for 3 years	$2,500	4	$10,000
Oracle Database Server Support Package for **3 years**	$2,000	1	$6,000
Oracle Mandatory E-Business Discount (license and support)			<$5,600>

Please let me know if you have any questions.

Thanks,
MaryBeth

1-Oct-2002

Dell Computer Corporation
Tony Kotecki
One Dell Way
Round Rock, Texas 78682
512.728.1207
tony_kotecki@dell.com

Kong:

The table below provides prices for Red Hat Advanced Server per your request for your TPC-R benchmarking testing. Please note that Dell provides Advanced Software Support when you purchase the Premier Enterprise Support Services bundled with you PE6600 Annual Premier Enterprise Support Services (PESS) contracts

The prices quoted here are for 24x7 coverage, 4-hour response for a period of 3-years. Per your instruction, I am including only the PESS 3-year with Red Hat Advance Server 2.1 no-factory installed prices.

Part Number	Description	Unit Price	Quantity	Total
420-1069	Red Hat Linux Advance Server 2.1	$2,499	1	**$2,499.00**
950-0117	PESS Gold for Advanced Software Support, 3 years	$ 300	1	**$300.00**
950-0128	PESS Gold 1st year Premium Services	$1,151	1	**$1,151.00**
950-0119	PESS Gold years 2 & 3 Premium Services	$666	1	**$666.00**
Total Cost for Services				**$4,616.00**

NOTE: support services include 30-day Help Getting Started for basic installation and configuration and product upgrade subscription through Red Hat Network for the first 3 years.
PESS Gold provides Advanced Software support including the Red Hat operating systems, as well as advanced hardware support bundled with the PowerEdge Server.
**SKU 420-1069 must be bundled & sold with a server; for a stand-alone operating system only, select SKU 310-1899 (same price, software only).*

This quote is valid for 90 days from date noted above.

Please feel free to contact me if I can be of any further help.

Thanks,

Tony Kotecki
Dell Product Marketing
Enterprise Services

PRICE QUOTE

Confidential and Restricted

October 11, 2002

Kong Yang
Dell Computer

Re: LSI Logic Elite 1650 RAID Controller Pricing Quote

Dear Kong,

LSI Logic is pleased to respond to your questions concerning the pricing quote pertaining to the MegaRAID Elite 1650 High Performance controller product.

We at LSI LOGIC greatly value the opportunity of maintaining our relationship as Dell's supplier of SCSI RAID controller products. Above all, we welcome the challenges of earning your trust and being the supplier most qualified to provide leading edge technology and impeccable customer support.

We are pleased of the opportunity to work closely with you and Dell, and look forward to your response of this pricing quote.

Should you have any questions or comments, please feel free to call me directly at 678-728-1271 or Elaine Morris-Winter at 678-728-1249.

Sincerely,

Jason Sabatino

Jason Sabatino
Product Marketing Engineer
Tel: 678-728-1271
Fax: 678-728-1214
Email: JasonS@lsil.com

This quotation is valid for 60 days from the date shown and is subject to the conditions as listed.

PRICE QUOTE

Confidential and Restricted

Date: 10/11/02

Company: Dell Computer
Contact Name: Kong Yang
Phone Number: 512-723-6051
Fax Number:

LSI LOGIC Elite 1650 High Performance MegaRAID Controller:

Item	Price per Unit	Description
Elite 1650 RAID Controller	$979.00	• 2 – Channel U160 PCI RAID Controller • 64MB Cache • High Performance Configuration

Conditions:

1) Quote is valid for 60 days from date received
2) Terms of sale to be determined prior to shipment

This quotation is valid for 60 days from the date shown and is subject to the conditions as listed.

Appendix I: Auditor's Attestation Letter

October 11, 2002

Greg Darnell & Kong Yang
Internet Performance
Dell Computer Corporation
One Dell Way
Round Rock, TX 78682

I have verified the TPC Benchmark™ R for the following configuration:

Platform: PowerEdge 6600/1.6 GHz
Database Manager: Oracle9i Enterprise Edition
Operating System: Red Hat Linux Advanced Server 2.1

CPU's	Memory	Total Disks	TPC-R Power @100GB	TPC-R Throughput @100GB	QphR@100GB
4 Intel Xeons MP @ 1600 Mhz	4 GB	2 @ 18GB, 10Krpm 4 @ 36GB, 10Krpm 84 @ 36GB,15Krpm	**12,802.6**	**1,548.5**	**4,452.5**

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

- The database tables were defined with the proper columns, layout and sizes.
- The tested database was correctly scaled and populated for 100GB using DBGEN. The version of DBGEN was 1.3.0.
- The qualification database layout was identical to the tested database except for the size of the files.
- The query templates were verified to use only compliant variants and minor modifications.
- The executable query text was generated by QGEN and submitted through "qexec" as the implementation specific layer. This program was verified to be compliant. The version of QGEN was 1.3.0.

- The random number seed was correctly initialized and incremented.
- The validation of the query text against the qualification database produced compliant results.
- The refresh functions were properly implemented and executed the correct number of inserts and deletes.
- The load timing was properly measured and reported.
- The execution times were correctly measured and adjusted.
- The performance metrics were correctly computed.
- The repeatability of the measurement was verified.
- The ACID properties were tested and verified.
- Sufficient mirrored log space was present on the tested system.
- The system pricing was checked for major components and maintenance.
- The executive summary pages of the FDR were verified for accuracy.

Auditor's Notes:

None.

Sincerely,

Lorna Livingtree
Auditor