Rdb Controller V6 For Oracle Rdb
(for Vax, Alpha, and Itanium)

User's Guide

March 2006

ALI Database Consultants

1151 Williams Dr.
Aiken, SC 29803 USA
Toll Free: (866) 257-8970
(803) 648-5931
Fax: (803) 641-0345
www.aliconsultants.com

DBAnalyzer for Oracle Rdb User's Guide for VAX/VMS
Version 1.0, February 1992
Version 2.0, May 1992
Version 2.0/2.2, October 1992
Version 2.5, August 1993
Version 2.6, October 1993
Version 3.0, September 1994

DBAnalyzer for Oracle Rdb User's Guide for Alpha AXP
Version 2.7, March 1994
Version 3.0, September 1994

DBAnalyzer for Oracle Rdb User's Guide (Combined Version)
Version 3.7, June 1995
Version 3.8, July 1995
Version 3.9, October 1996
Version 5.0, June 1997
Version 5.2, November 1998
Version 5.2a, October 2000
Version 5.2a/5.3, September 2001
Version 5.2a/5.3, June 2002
Version 5.2a/5.3, August 2002
Version 5.2a/5.3, September 2002
Version 5.2a/5.3, May 2003
Version 6.0, March 2006

DBTune for Oracle Rdb User's Guide for VAX/VMS
Version 1.0, May 1992
Version 2.0/2.2, October 1992
Version 2.5, August 1993
Version 2.6, October 1993
Version 3.0 (FT2), July 1994
Version 3.0, September 1994

DBTune for Oracle Rdb User's Guide for Alpha AXP
Version 2.7, March 1994
Version 3.0, September 1994

DBTune for Oracle Rdb User's Guide (Combined Version)
Version 3.5, February 1995
Version 3.7, June 1995
Version 3.8, July 1995
Version 3.9, September 1996
Version 4.0, June 1997
Version 5.0, June 1997
Version 5.1, March 1998
Version 5.2a, October 2000
Version 5.2a/5.3, September 2001
Version 5.2a/5.3, June 2002
Version 5.2a/5.3, August 2002
Version 5.2a/5.3, September 2002
Version 5.2a/5.3, May 2003
Version 6.0, March 2006

DBXAct for Oracle Rdb User's Guide for VAX/VMS
Version 1.0, December 1993
Version 1.2, January 1994
Version 1.3, February 1994
Version 1.4, March 1994
Version 3.0 (FT2), October 1994
Version 3.5, January 1995

DBXAct for Oracle Rdb User's Guide for Alpha AXP
Version 1.0, December 1993
Version 1.2, January 1994
Version 1.3, February 1994
Version 1.4, March 1994
Version 3.0 (FT2), October 1994
Version 3.5, January 1995

DBXAct for Oracle Rdb User's Guide (Combined Version)
Version 3.7, June 1995
Version 3.8, July 1995
Version 3.9, March 1996
Version 5.0, March 1998
Version 5.1, April 1999

Version 5.1/5.2, September 2001
Version 5.1/5.2, August 2002
Version 5.2a/5.3, September 2002
Version 5.2a/5.3, May 2003
Version 6.0, March 2006

NOTE: This manual applies to Rdb Controller 6.0 for I64, Vax and Alpha AXP running Rdb 6.1 or higher

Acknowledgments

The development team at ALI would like to thank the following consultants and clients who have taken time to review our products and make suggestions for improving our products. Rdb Controller for Oracle Rdb has many features that were suggested by these people. We regret that we may not have been able to implement all suggestions for this release, but we try to shape each release as closely as possible to clients' needs and believe that the success of our products depends on our ability to incorporate improvements that users request.

COMMONWEALTH ALUMINUM	Jean Dickens
ITSD	Richard Dean
NOVOPHARM	Mohammed Aziz
SAFILO	Jeff Wolfe
SEA CONTAINERS (UK)	Kevin Gordon
VNU	Latesha Williams

Contents

Introduction

What is Rdb Controller for Oracle Rdb?

Rdb Controller for Oracle Rdb

Rdb Controller™ provides a solution for many Oracle Rdb environments by analyzing, monitoring, and tuning the entire database system to identify problem areas before they impact the user community. Rdb Controller not only properly tunes the storage areas, indexes, table distribution, and I/O distribution, it performs these actions in record time. Tuning efforts can be reduced drastically, and performance improvements typically range from 20 to 50 percent.

Additionally, to ensure your Oracle Rdb environment is constantly performing as expected, Rdb Controller for Rdb provides you with the ability to continuously monitor Oracle Rdb and identify problem areas before they impact your user community.

Real-Time Monitoring

By using the DBXact feature in Rdb Controller for Rdb, users can automatically monitor their Rdb database and identify those tables, indices, and areas that are the most heavily used, and which are exhibiting performance degrading behavior (locking, stalls, etc).

DBXact checks your database, at user defined intervals, to determine the busiest storage areas/tables/indices and disk drives, and identifies the database growth. In addition to reports detailing the observed activities, a separate activity file can be generated to automatically input the observed activities into the tuning script generation portion of the product (DBTune).

Minimizing Fragmentation

By using the DBTune feature in Rdb Controller for Rdb, users can automatically implement a database tuning solution to resolve fragmentation problems. DBTune partitions the database via storage areas, intelligently distributes storage areas across multiple disk drives, and appropriately sizes storage areas to handle growth within the database. This enables you to proactively tune the database, so your database will perform optimally for months.

Rdb Controller for Rdb first analyzes your database activity to determine the busiest storage areas and disk drives, and identifies the growth level for the storage areas. It also analyzes where to distribute tables and indexes, so that user access times are minimized. Then, Rdb Controller generates all the necessary tuning scripts to tune the entire database, or a limited number of specific storage items. These scripts can take days to generate manually, where Rdb Controller for Rdb can complete them in minutes and ensure all dependencies and steps are in place with error checking for the tuning process.

Rdb Controller for Rdb provides a complete tuning process implemented with a DCL driver that runs the generated SQL scripts to perform the tuning. These scripts can be executed to tune your entire database, or simply the most important areas, depending on the maintenance window you have available. ALI's customers typically experience a 20 to 50 percent improvement in application performance.

Choose the Right Indexes

Rdb Controller for Rdb reviews indexes and database activity to identify conversion considerations. It provides recommendations to change the index type to either SORTED, or HASHED based on likely access approaches. The decision should be made based on the actual retrieval and concurrency needs of the database users. Rdb Controller implements all necessary work to modify SORTED, SORTED RANKED, HASH SCATTERED, and HASH ORDERED indexes and structure them for optimal performance. It generates a SQL procedure that will change the database and tune it accordingly.

Minimizing Locking Occurrences

Rdb Controller for Rdb can assist with these advanced tuning requirements in several ways: by locating hashing opportunities and providing record counts, record sizes, and index information to calculate appropriate page size and allocation.

Rdb Controller for Rdb can be used to implement and properly tune for clustering and shadowing.

I/O Distribution

Rdb Controller for Rdb automatically balances I/O for individual storage areas, or the entire database, to ensure your users have the quickest access to data. It identifies which tables are the most active, sizes their storage areas accordingly, distributes them across separate disk drives, and follows the rules of thumb for I/O listed below.

- Do not place all your indexes on a single disk drive when not using large RAID units with much built-in caching.
- Separate the table and index if you are not using the **PLACEMENT VIA** option.
- Keep RDA and SNP files on separate disks.
- Keep your busiest storage areas on separate disks.

Features of the Products within Rdb Controller for Rdb

DBAnalyzer for Rdb

- Scans a database and performs a summary analysis.
- Provides four views of a database: a **MACRO** view, a **MICRO-TABLE** view, a **MICRO-INDEX** view, and a **MICRO-STORAGE AREA** view.
- Provides overall and individual statistics for databases.
- Produces organized output that provides you with a comprehensive view of a database and gives some measure of its tuning status.

DBTune for Rdb

- Creates a Performance Analysis Data file, using the logical and physical data gathered from the database. The Performance Analysis Data file contains volume, workload, and environment information, which you may customize to include additional transaction activity.
- Maximizes Rdb performance and maintenance without requiring excessive effort on your part.
- Separates larger tables and the associated indexes into their own tablespaces for ease of monitoring and load distribution to improve performance.
- Tunes the entire physical structure of the database, typically resulting in a 50+ percent improvement in application performance.

DBXAct for Rdb

- Monitors performance of the database and generates baseline statistics and reports on Rdb activity, making it possible for you to clearly understand the hundreds of data points available for monitoring database activity.

- Produces detailed statistical reports that allow you to perform precise tuning and optimization of the Rdb database.

- Establishes benchmarks and presents the results of the tuning changes in order to determine if the tuning efforts have been successful.

- DBXAct produces a database activity file that can be used by DBTune to evaluate the tuning needs of the database and create SQL scripts for its physical redesign. The database activity file provides you with the information needed to successfully tune and optimize the Rdb database.

Chapter 1

DBAnalyzer for Rdb

What is DBAnalyzer?

The DBAnalyzer procedure scans a database pointed to by the logical ALI_RDB_IMPORT. During this scan, the procedure performs a summary analysis, which is then displayed on your screen or written to an output file. If called in **BATCH** mode, the procedure will create the output file only.

DBAnalyzer provides four views of a database: a **MACRO** view, a **MICRO-TABLE** view, a **MICRO-INDEX** view, and a **MICRO-STORAGE AREA** view.

The **MACRO** view provides overall statistics for the database—the five largest tables, the five indexes with the most duplicates, etc. The purpose of these **MACRO** windows is to highlight likely hot spots in the database that may require tuning attention.

The **MICRO-TABLE** view provides statistics for individual tables/views in the database. You may scroll (alphabetically) through the tables/views or jump directly to a particular table by entering a search string (to which the table names are compared).

The **MICRO-INDEX** view provides statistics for individual indexes in the database. Here again, you may scroll through indexes (based on their associated tables) or jump directly to a specific table by entering a table name search string.

The **MICRO-STORAGE AREA** view provides statistics for each storage area in the database. As with the other MICRO views, you may scroll through each storage area or jump directly to a specific storage area by entering the storage area search string.

Output can also be obtained from this procedure and several options are provided. The default name for the output file is DBANALYZR.LST, but you can override this by typing in a different file name when prompted. See the DBAnalyzer Reports section.

Note DBAnalyzer 6.0 works with Rdb higher than 6.1

Tune and Complexity Ratings

The purpose of DBAnalyzer is to produce organized output that provides you with a comprehensive view of a database and give some measure of its tuning status. Two indicators of a database's tuning status are the TUNE RATING and the COMPLEXITY RATING values. The TUNE RATING is a measure of the extent to which existing tuning options have been utilized in the database.

The TUNE RATING does not directly indicate the percentage increase in performance that can be achieved by tuning a database. Rather, the impact of tuning is more directly related to the complexity of the database. Therefore, the TUNE RATING in conjunction with the COMPLEXITY RATING is a more complete measure of a database's tuning status and the increase in performance that can be achieved by further tuning.

The INTEGRITY RATING of the database is a measure of how effectively the database has implemented constraints. The measurements that comprise this INTEGRITY RATING are:

1. Columnar Integrity Rating

2. Referential Integrity Rating

3. Referential Efficiency Rating

These individual components can be viewed on MACRO screen 14 or by displaying and/or printing the DBAnalyzer narrative. This narrative is available online by pressing the **Find** key, or in batch as part of the DBAnalyzer report. The narrative explains the meanings of the various graphs and ratings that are presented in DBAnalyzer.

Getting Started

This section provides the information needed for you to quickly install and run DBAnalyzer. It also includes system requirements necessary to run the application.

DEC VAX/OpenVMS

Recommended **minimum AUTHORIZE** settings for a DBAnalyzer user account. (Medium and large databases may need to increase these numbers.)

Username:	USER			Owner:	USER
Account:	USER			UIC:	[Group, Member]
CLI:	DCL			Tables:	DCLTABLES
Default:	<disk>:[dir]				
LGIMD:	LOGIN				
Login Flags:					
Primary Days:	Mon. Tues. Wed.	Thurs. Fri.			
Secondary Days:	Sat. Sun.				
No access restrictions					
Expiration:	(none)	Pwdminimum:	0	Login Fails:	0
Pwdlifetime:	(none)	Pwdchange:			
Last Login:					
Maxjobs:	0	Fillm:	512	Bytlm:	90000
Maxacctjobs:	0	Shrfillm:	100	Pbytlm:	0
Maxdetach:	0	BIOlm:	100	Jtquota:	1024
Prclm:	4	DIOlm:	100	Wsdef:	1024
Prio:	4	ASTlm:	113	Wsquo:	1024
Queprio:	0	TQElm:	10	Wsextent:	1024
CPU:	(none)	Enqlm:	8000	Pqlfquo:	80000
Authorized Privileges:	GROUP TMPMBX NETMBX				
Default Privileges:	GROUP MPMBX NETMBX				

Warning If these minimums are not in place when DBAnalyzer is executed, the analysis may fail!

DEC AXP/OpenVMS & I64/OpenVMS

Recommended **minimum AUTHORIZE** settings for a DBAnalyzer user account. (Medium and large databases may need to increase these numbers.)

Username:	USER			Owner:	USER
Account:	USER			UIC:	[Group, Member]
CLI:	DCL			Tables:	DCLTABLES
Default:	<disk>:[dir]				
LGIMD:	LOGIN				
Login Flags:					
Primary Days:	Mon. Tues. Wed. Thurs. Fri.				
Secondary Days:	Sat. Sun.				
No access restrictions					
Expiration:	(none)	Pwdminimum:	0	Login Fails:	0
Pwdlifetime:	(none)	Pwdchange:			
Last Login:					
Maxjobs:	0	Fillm:	512	Bytlm:	90000
Maxacctjobs:	0	Shrfillm:	0	Pbytlm:	0
Maxdetach:	0	BIOlm:	150	Jtquota:	1024
Prclm:	4	DIOlm:	150	Wsdef:	2000
Prio:	4	ASTlm:	250	Wsquo:	4096
Queprio:	0	TQElm:	10	Wsextent:	16384
CPU:	(none)	Enqlm:	8000	Pqlfquo:	80000
Authorized Privileges:	GROUP TMPMBX NETMBX				
Default Privileges:	GROUP MPMBX NETMBX				

Warning If these minimums are not in place when DBAnalyzer is executed, the analysis may fail!

Installing DBAnalyzer

To install DBAnalyzer for Rdb from a tape drive:

1. Back up your system disk (optional).
2. Log in under the SYSTEM account.
3. Put the DBAnalyzer distribution tape in the tape drive.
4. Type in the following command to invoke the VMS install facility to install DBAnalyzer on your system:

   For VAX/VMS

   ```
   $ @SYS$UPDATE:VMSINSTAL DBARDBVMS060 <<tape-drive>>:
   ```

   For Alpha AXP

   ```
   $ @SYS$UPDATE:VMSINSTAL DBARDBAXP060 <<tape-drive>>:
   ```

   For Itanium I64

   ```
   $ @SYS$UPDATE:VMSINSTAL DBARDBITA060 <<tape-drive>>:
   ```

   where <<tape-drive>> is the name of the device where the DBAnalyzer distribution tape has been mounted (e.g., MUA6:).

> **Note** DBAnalyzer 6.0 should NOT be installed in the same directory with other versions of DBAnalyzer or any other product from ALI (i.e., DBTune).

5. After the VMS install has completed, place the following lines into the system startup command file

   (SYS$MANAGER:SYSTARTUP_VMS.COM) so that required logicals are set up when the system is rebooted:

   ```
   $ DEFINE/SYSTEM/EXEC ALI_DBA_HOME       <<disk>>:[dir]
   ```

```
$ DEFINE/SYSTEM/EXEC ALI_DBA_SCRATCH
   <<disk>>:[dir.scratch]
```

where <<disk>> and [dir] are the disk and directory to which DBAnalyzer was installed (e.g.,
1DUA1:[DBARDBVMS60.SCRATCH] or
1DUA1:[DBARDBAXP60.SCRATCH] or
1DUA1:[DBARDBITA60.SCRATCH]).

6. Now, to obtain a license pak for DBAnalyzer, type in the following commands:

```
$ SET DEFAULT ALI_DBA_HOME

$ EDIT DBANLZR.LICENSE
```

For each node ("machine") on which you wish to run DBAnalyzer:

- Replace "your node name" with the node name of the machine on which you have installed DBAnalyzer. To get this information, type:

```
$ WRITE SYS$OUTPUT F$GETSYI ("nodename")
```

- If the "operating system" value supplied with your license is not accurate for your system, replace it with the output generated from the following command:

```
$ WRITE SYS$OUTPUT F$GETSYI ("node_swtype")
```

- Replace "your company name" with your company's full name
- Exit and save the file

To obtain the appropriate registration ID for each machine entered, call ALI at (866) 257-8970 [or (803) 648-5931], or fax a copy of the altered DBANLZR.LICENSE file to (803) 641-0345. International clients may also obtain registration IDs/support through their local distributor's office.

To install DBAnalyzer for Rdb from a CD-ROM:

1. Mount the CD using a command like

   ```
   $ MOUNT/OVER=ID <cd_device>:
   ```

2. Install the product with the command

   ```
   $ @SYS$UPDATE:VMSINSTAL <product_name>
   <cd_device>:[INSTALL]
   ```

   where <product_name> is the product you wish to install.

   For example:

   ```
   $ @sys$update:vmsinstal DBARDBVMS060 dka400:[INSTALL]
   ```

Using DBAnalyzer

DBAnalyzer may be executed either ONLINE or in BATCH. If executed ONLINE, DBAnalyzer is invoked by the command file DBA.COM. If executed in BATCH, DBAnalyzer is invoked by submitting the command file DBA_BATCH.COM. Before using DBAnalyzer, however, check with your system manager—VMS symbols may have been set up to facilitate use of this facility:

e.g., **`DBA:==@ALI_DBA_HOME:DBA.COM`**

There is one logical that is required to be assigned in order to execute DBAnalyzer in either ONLINE or BATCH mode:

`ALI_RDB_IMPORT`

DBAnalyzer scans the Rdb database pointed to by the logical ALI_RDB_IMPORT and expects this logical to be assigned prior to execution. You must also have privilege to access the database that you chose to analyze.

Online Use of DBAnalyzer

If executed ONLINE, the DBAnalyzer utility will check the assignment of ALI_RDB_IMPORT. If not assigned, you will be prompted for the location and name of the Rdb database to be analyzed. If the logical is already pointing to a database, however, you will be asked if you would like to point to a different database before continuing.

To invoke DBAnalyzer online:

- If the symbol "DBA" has been created, type the following:

```
$ DBA
```

You will receive the following prompt:

The logical ALI_RDB_IMPORT, which must be assigned to the database that you wish to analyze, is currently unassigned. If you wish to continue, type in the disk, directory, and name of the database you wish to analyze or press Ctrl-Z to quit.

Example: `DISK1:[MYDATA.RDB]PERSONNEL.RDB`

Specify a DATABASE: `DISK5:[ACCTNG.RDB]INVOICE.RDB`

- If no symbol exists for DBAnalyzer, type the following:

```
$ @ALI_DBA_HOME:DBA.COM
```

Batch Use of DBAnalyzer

If executed in BATCH, DBAnalyzer expects the ALI_RDB_IMPORT logical to have been assigned prior to execution—you will not be prompted. To this end, a command file has been provided to allow assignment of ALI_RDB_IMPORT. This command file—DBA_BATCH.COM—can be edited to select the database that will be scanned.

```
$!------------------------------------------------------------------------
$!      DBA_BATCH.COM
$!        - Command file to submit DBAnalyzer in BATCH mode...
$!
$!      Invoke this file with the command:
$!      $ @ALI_DBA_HOME:DBA_BATCH
$!------------------------------------------------------------------------
$!
$!      This command procedure will submit itself to batch.
$!
$!       if p1 .eqs. "" .or. p1 .nes. "BATCH"
$!           then
$!
$!      Change the /name="" qualifier to specify a different name for the job.
$!
$!               cur_def = f$environment("DEFAULT")
$!               vfl = f$verify(0)
$!               set verify
$!               submit-
                         /log-
                         /noprint-
                         /name="DBA in Batch"-
                         /parameters=("BATCH","''cur_def'") -
                         ALI_DBA_HOME:DBA_BATCH.COM
$!               vfl = f$verify(vfl)
$!               exit
$!           endif
$!-------------------------------------------------------------------------
$!       Change the following ASSIGN statement to point the database you wish to
$!       analyze and uncomment it by removing the "!" ...
$!
$!       ASSIGN "disk1:[directory]database_name" ALI_RDB_IMPORT
$!
$!               Change the following SET PROC/NAME= to assign a different
$!               process name and uncomment it by removing the "!" ...
$!
$!       SET PROC/NAME="DBA in Batch"
$!
$!               Change the following SET DEFAULT to change the default
$!               directory where the report will be generated. Otherwise,
$!               output will be generated the user's current directory at the
$!               time the file was submitted.
$!
$!       SET DEFAULT 'p2'
$!------------------------------------------------------------------------
$!       @ALI_DBA_HOME:DBA.COM
```

To invoke DBAnalyzer in batch:

- Type the following:

```
$ @ALI_DBA_HOME:DBA_BATCH.COM
```

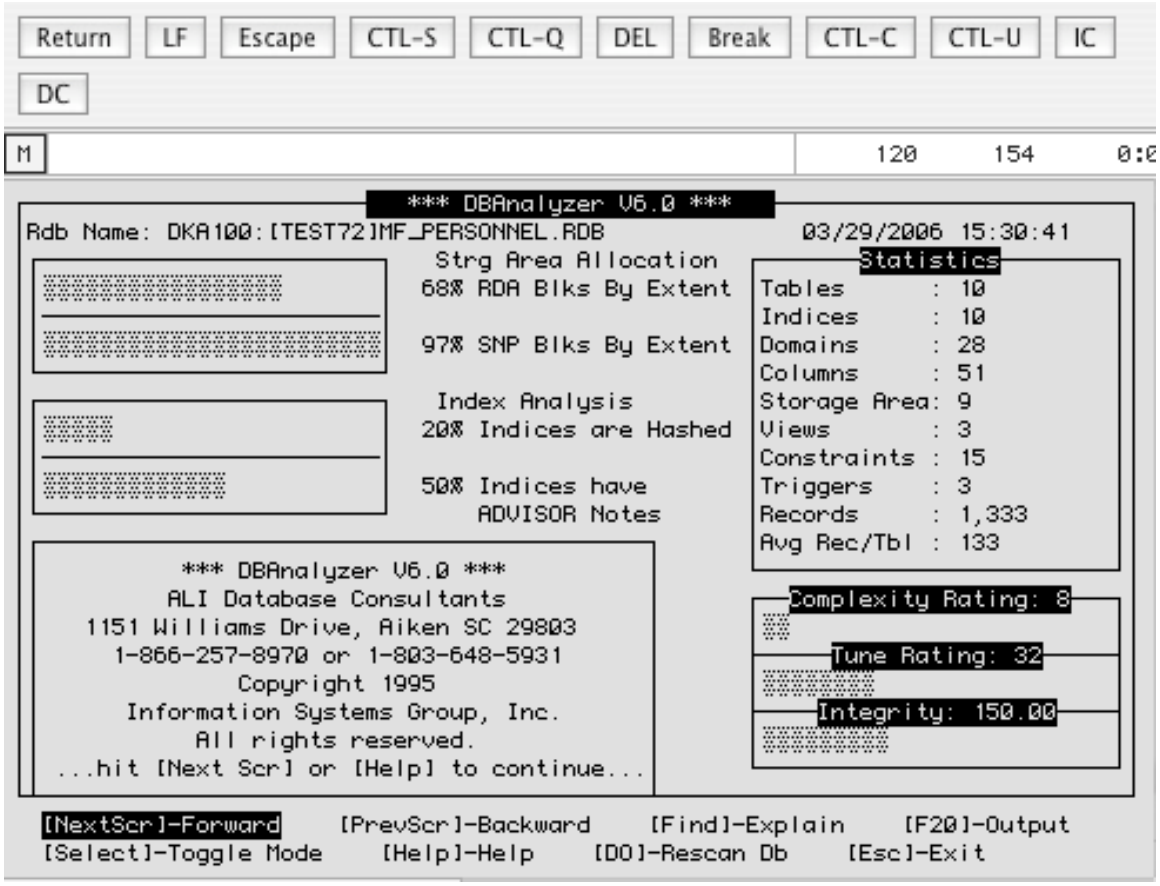
You can optionally set up a customized report parameter file that can be used to govern the format of the output report created when DBAnalyzer is executed in BATCH. You can specify a customized report parameter file for DBAnalyzer by assigning the logical ALI_DBA_PARAM_FILE and pointing it to a valid parameter file. This assignment can be made in the DBA_BATCH.COM file mentioned previously.

On the following page is an example of such a parameter file with valid entries and default values listed.

Example of an optional DBAnalyzer report parameter file

```
                                                Defaults
dba.report.name:  MYFILE.RPT                    [DBANALYZER.LST]
*
*       Selection Criteria
*
dba.select.full:  YES or NO                     [YES]
dba.select.brief:  YES or NO                    [NO]
dba.select.domains:  YES or NO                  [YES]
dba.select.narrative:  YES or NO                [YES]
dba.select.storage_areas:  YES or NO            [YES]
dba.select.tables:  YES or NO                   [YES]
dba.select.views:  YES or NO                    [YES]
*
*       Printer Criteria
*
dba.print_report:  YES or NO                    [NO]
dba.printer: SYS$PRINT or blank                 [  ]
dba.printer_options: {/FORM=LETTER16/HEADER}    [  ]
*
*       Domain Sort Order Criteria
*
dba.domain.sort.order: DOMAIN, COLUMN, TABLE    [DOMAIN]
*
*       Storage Input Criteria
*
dba.storage_source:  ALL or FILE                [ALL]
*
*       Table Detail Criteria
*
dba.table_source:  ALL or FILE                  [ALL]
dba.table_option:  FULL or BRIEF                [FULL]
dba.table_columns:  YES or NO                   [YES]
dba.table_indices: YES or NO                    [YES]
dba.table_indices_option:  FULL or BRIEF        [FULL]
*
*       View Detail Criteria
*
dba.view_source:  ALL or FILE                   [ALL]
dba.view_option:  FULL or BRIEF                 [FULL]
```

Example of DBAnalyzer's Online Display



DBAnalyzer Keystrokes

The following keystrokes may be used when executing DBAnalyzer ONLINE using a VT220 (or higher) terminal interface:

Help or PF2	Receive help for the current DBAnalyzer context. Repeated execution will allow viewing of the entire help menu
Esc PF3 PF4	Exit DBAnalyzer or return to MACRO mode
Do	Rescan the Database and recalculate analysis information
Select	Toggle from MACRO mode to MICRO-TABLE mode to MICRO-INDEX mode to MICRO-STORAGE AREA mode
Next Screen	Scroll to next window
Prev Screen	Scroll to previous window
Find	Display Narrative [MACRO window] Display View [MICRO table window]
F20	Write report to an output file
Ctrl W	Refresh screen display

DBAnalyzer Report Generation Keystrokes

The following keystrokes may be used when executing the DBAnalyzer report generation functions using a VT220 (or higher) terminal interface. (See also the DBAnalyzer Reports section on page 33.)

PF3	Exit back to SCAN mode
PF4	Backup one window (not implemented in this release)
Select	Highlight an option for execution
Return	Execute options
	Move up in selection window
	Move down in selection window
Next Screen	Page down in a selection window
Prev Screen	Page up in a selection window
Help	Help on Report Generation

DBAnalyzer Process

The following information is available from DBAnalyzer:

Rdb Statistics

Total number of Tables, Indexes, Storage Areas, etc., as well as database parameters such as Global Buffers, Buffer size, Number of Users, etc.

Complexity Rating

The **COMPLEXITY RATING** is a weighted measure of the size and complexity of a database. The **NARRATIVE** analysis provides a relative description of its significance.

Tune Rating

The **TUNE RATING** is a measure of the extent to which existing tuning options have been utilized in the database. A database with a **TUNE RATING** of between 80 to 100 is considered one that has taken advantage of available tuning options. A rating of zero suggests that no tuning (other than the provided defaults) has been performed.

The **TUNE RATING** does not directly indicate the percentage increase in performance that can be achieved by tuning a database. Rather, the impact of tuning is more directly related to the complexity of the database. Therefore, the **TUNE RATING** in conjunction with the **COMPLEXITY RATING** is a more complete measure of a database's tuning status and the increase in performance that can be achieved by further tuning.

The **TUNE RATING** indicates how well the physical design supports the current logical design. The rating is a ratio of storage area utilization (factored for allocation efficiency) and index design compared to the database complexity. The **NARRATIVE** analysis provides an explanation of the performance impact of the **TUNE RATING** for the given **COMPLEXITY**.

Integrity Rating

The **INTEGRITY RATING** of the database is a measure of how effectively the database has implemented constraints. The measurements that comprise this **INTEGRITY RATING** are:

1. Columnar Integrity Rating
2. Referential Integrity Rating
3. Referential Efficiency Rating

A database with an **INTEGRITY RATING** between 75 and 100 is considered to be one that has taken advantage of Rdb's constraint mechanisms to ensure the completeness and integrity of data. DBAnalyzer's **NARRATIVE** analysis provides a detailed description for a particular database.

Storage Area Allocation

The **STORAGE AREA ALLOCATION** consists of two graphs that show the percentage of blocks that have been created due to extensions. The graphs indicate the percentage for both the RDA and SNP files. These ratios are used to factor the storage area utilization portion of the **TUNE RATING**. That is, the higher the allocation percentages, the lower the tune rating.

Hashed Index Percentage

The **HASHED INDEX PERCENTAGE** is the ratio of hashed indexes to the total number of indexes in the database. If you have ten indexes of which two are hashed, then the **HASHED INDEX PERCENTAGE** would be 20 percent.

HASH-to-SORT/SORT-to-HASH Ratio

HASH-to-SORT/SORT-to-HASH RATIO is the ratio of indexes that have ADVISOR recommendations to be changed from either hashed to sorted or sorted to hashed. If you have ten indexes of which two sorted indexes are recommended to be hashed and one hashed index is recommended to be sorted, then the HASH-to-SORT/SORT-to-HASH RATIO would be 30 percent.

Macro View

The MACRO VIEW consists of 14 windows to provide statistics on significant items from various aspects of the database. Each of the 14 windows is described below. These windows are not intended to be a comprehensive list of all of the items that should be tuned. Rather, they are listed as ones that are likely to have significant tuning implications relative to other items in the database. Remember that overall tuning benefits available are relative. The expected tuning benefits are impacted by numerous factors. DBAnalyzer cannot predict exact improvements. You can, however, expect tuning benefits to have a direct relation to the COMPLEXITY RATING and an indirect relation to the TUNE RATING. That is, the higher the COMPLEXITY RATING and the lower the TUNE RATING, the more the database performance may be improved through tuning.

- Macro Window 1:
  Tables with the Highest Record Counts

  The five tables with the highest record counts (cardinality) are listed in descending order. These tables are likely candidates for special tuning attention.

- Macro Window 2:
  Indexes with the Most Duplicates

  The five indexes with the highest average number of duplicates are listed in descending order. Indexes with numerous duplicates can unnecessarily increase append, modification, and deletion processing time.

- Macro Window 3:
  Tables with the Most Columns

  The five tables with the most columns are listed in descending order. These tables may indicate the presence of redundant fields or fields that can be more efficiently stored in other tables. Rdb has clustering and shadowing capabilities that can potentially improve performance by greater normalization rather than reduced normalization. These capabilities often apply to one-to-many relationships.

- Macro Window 4:
  Tables with the Largest Record Size

  The five tables whose columns require the most bytes to store a 'full' record. Although Rdb stores each column as its own item, proper page sizing techniques make use of this effective record size.

- Macro Window 5:
  SORTED Indexes Recommended to be HASHED

  The first five SORTED indexes that DBAnalyzer encounters that it determines may be a candidate for Hashing. HASHED indexes may be used to decrease the number of I/Os required to retrieve a record and increase concurrency by reducing the number of required locks. Additional benefits may accrue when hashed indexes are used in conjunction with CLUSTERING and SHADOWING.

- Macro Window 6:
  HASHED Indexes Recommended to be SORTED

  The first five HASHED indexes that DBAnalyzer encounters that it determines may be a candidate for Sorting. SORTED indexes may be necessary to decrease the number of I/Os required for retrieving a range of records. An example of this type of retrieval is gathering all employees with a last name starting with 'SM'. Sorted indexes may reduce concurrency for records whose index values fall into the same index NODE. The effect of such occurrences depends on the values being accessed and the mode in which they are accessed.

- Macro Window 7:
  Tables with the Most Indexes

  The five tables with the most indexes are listed in descending order. The indexes for these tables should be reviewed to see if they are being used. Each index must be maintained whenever records are added, modified, or deleted. Indexes require disk space, and they use resources for their maintenance. Indexes that are not used should be removed.

- Macro Window 8:
  Non-Indexed Tables with the Most Records

  The five tables that do not have any indexes are listed in descending order by record count. Retrieval of data from these tables requires that all of the records in the table be retrieved. Performance may be improved if an index can be created that will allow the Rdb-Optimizer to select a smaller set of these tables' records.

- Macro Window 9:
  Storage Areas with the Most Mapped Items

  The storage areas, into which the most items are mapped, are listed in descending order by mapped item count. The mapped item can be a table, sorted index, or hashed index. These areas may not perform as well as other areas since multiple types of data must be stored in the same area. Thus, performance may not be as efficient as possible. The performance impact increases as the number of records increases for each of the items.

- Macro Window 10:
  Storage Areas with the Most File Extensions

  The five RDA files that have extended the most times are listed in descending order. File extensions occur when there is insufficient space to hold new data. File extensions may indicate insufficient pages, incorrect page sizes, or both.

- Macro Window 11:
  RDA Files with the Most Extension Blocks

  The five RDA files that have the most extension blocks are listed in descending order. File extensions occur when there is insufficient space to hold new data. File extensions may indicate insufficient pages, incorrect page sizes, or both.

- Macro Window 12:
  SNP Files with the Most Extension Blocks

  The five SNP files that have the most extension blocks are listed in descending order. File extensions occur when there is insufficient space to hold new data. SNP files are used to hold "snapshots" of pages that are locked, so that READ transactions can access these pages. Extended space indicates that the snapshot activity requires more space than that initially allocated.

- Macro Window 13:
  Database Storage Area Extension Summary

  The storage area extension summary displays the total number of RDA and SNP blocks that were initially allocated and how many are currently in use. Additionally, the total number of RDA extensions is available.

- Macro Window 14:
  Integrity Rating Constituent Components

  The integrity rating constituent components display a graphical and numeric representation of the three components of the database's composite integrity rating.

- Macro Window 15:
  Database Recovery Summary

  This display shows the current state of after-image journals, recovery buffers, and the time of the last backup.

- Macro Window 16:
  System Information

  This window displays OpenVMS information that is applicable to the operation and performance of the database, icluding total number of running processes, % cpu busy, and available/free memory.

- Macro Window 17:
  Disks with Least Space

  This display shows the current database disks which have the least amount of free space.

Micro-Table View

The MICRO-TABLE VIEW is presented by pressing the **Select** key while in MACRO MODE. When you select **MICRO-TABLE MODE**, you are will see the first non-system table in the database, in alphabetic order by table name. You may press the **Next Screen** key to progress through the database tables alphabetically. You may also type a table name or a portion of a table name and press **Return**. DBAnalyzer will locate the first table name that is greater than or equal to the entered letters. When DBAnalyzer locates the end of the list of tables, it returns to MACRO MODE.

MICRO Mode: Rdb Tables/Views

Table: CUST_CONTACTS	
Cardinality (Record Count):	342
Number of Columns:	6
Number of Bytes (Record Size)	87
Storage Area: CUST_CONTACT_AREA	
Number of Indexes:	1
1st Unique Index: CONTACT_TYPE	

The Micro-Table window presents critical tuning information for each non-system table in the database. The number of columns and their byte allocations are used to calculate proper page sizes, and the number of records indicates how many pages to allocate. Proper page size and allocation prevent fragmentation within the storage area and prevent fragmenting extents for the storage area file itself. The first storage area used by the table is listed. RDB$SYSTEM is the default storage area. Those tables with the most records should have priority when deciding which tables to move into their own storage area.

The number of indexes is a quick check on possible database performance problems. A high number of indexes may be wasteful and inefficient, but no indexes may require extra I/O, especially for tables with a large number of records and frequent retrievals of a subset of the table records. The proper number of indexes is a trade-off in the processing time to maintain an index versus the savings from index-based retrieval.

VIEWs may be displayed in the MICRO-Table view by using the **Find** key. The display shows the columns and the selection that comprise the view.

Note The DBAnalyzer report shows all storage areas and indexes for a table. See the DBAnalyzer Reports section for more information.

Micro-Index View

The MICRO-INDEX VIEW is presented by pressing the **Select** key while in MICRO-TABLE MODE. When you select **MICRO-INDEX MODE**, you will see the first index for the first non-system table in the database, in alphabetic order by table and the table's index. You may press the **Next Screen** key to progress through the database indexes. You may also type a table name or a portion of a table name and press **Return**. DBAnalyzer will locate the first index for the first table name that is greater than or equal to the entered letters. When DBAnalyzer locates the end of the list of indexes it returns to MACRO MODE.

```
                MICRO Mode:  Rdb Indexes


Table          :  CUST_CONTACTS
Index (001)    :  CONTACT_TYPE
Index Type     :  SORTED, NON-UNIQUE
Avg Dups       :  27
Storage Area   :  CONTACT_TYPE_IDX_AREA
Index columns:    1
1st column     :  CUST_CONTACT_TYPE
```

The Micro-Index window presents information about a selected index. It shows the table to which the index belongs, whether it is Sorted or Hashed, the first storage area it is mapped into, the number of columns, and the first column in the index. The window allows a DBA to quickly see what indexes are available for record retrieval.

Those indexes with a high number of average duplicates are ones that should be reviewed. Unless specific techniques such as clustering and shadowing of records are being employed, indexes with a high number of average duplicates may be creating extra processing. Adding columns to the index may reduce the number of average duplicates, index processing and improve performance.

Note The DBAnalyzer report shows all storage areas and columns for an index. See the DBAnalyzer Reports section for more information.

Micro-Storage Area View

The MICRO-STORAGE AREA VIEW is presented by pressing the **Select** key while in MICRO-INDEX MODE. When you select **MICRO-STORAGE AREA MODE**, you will see the first storage area in the database, in alphabetical order by storage area name. You may press the **Next Screen** key to progress through the database storage areas. You may also type a storage area name or a portion of a storage area name and press **Return**. DBAnalyzer will locate the first storage area that is greater than or equal to the entered letters. When DBAnalyzer locates the end of the list of storage areas, it returns to MACRO Mode.

```
            MICRO Mode:  Rdb Storage Areas


      Storage Area: COMP_NAME_INDEX_AREA
      Page Size:   4 blocks, Format:  UNIFORM
      Number of Extents:                          1
      RDA Extensions (blocks):                  194
      SNP Extensions (blocks):                    0
      ------------Stored Elements-----------
            Tables:   0   Sorted:   1   Hashed:  0
```

The Micro-Storage Area window presents information on the initial and current pages for both the RDA and SNP files. Additionally, the number of each type of mapped item is displayed. Large storage areas may achieve better performance if only "related" items are clustered together. For example, clustering the HASH index and its table may achieve single I/O performance when the HASH key is used to retrieve the table.

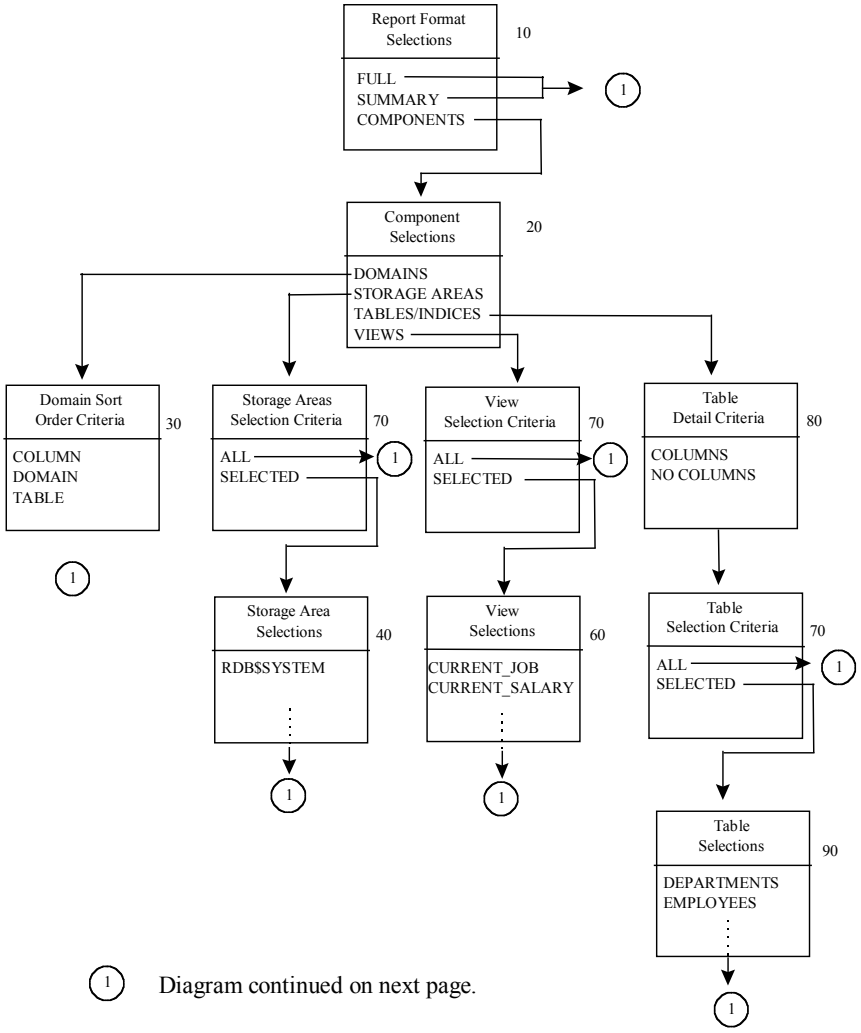
DBAnalyzer Reports

Report generation for DBAnalyzer has been modified in order to give you more flexibility in the type of report you generate.

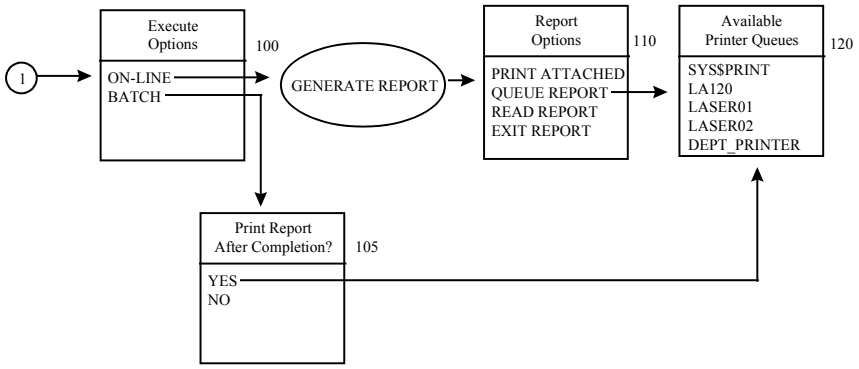
DBAnalyzer provides a window environment for you to select the flavor of your report. **FULL** and **SUMMARY** report options are available, and with the window interface, you may specify a report to provide information on **DOMAINS**, **STORAGE AREAS**, **TABLES**, and **VIEWS**. These may be done exclusively of one another or combined together in one report.

In addition, you can generate variations on the **/Table**, **/Index**, and **/Storage Area** reports. The **STORAGE AREAS** report is tabular in orientation and reports **TABLES** and **VIEWS** independently of one another to more logically segregate your data needs. The **DOMAIN** report will provide you with a means of viewing your columns/tables in any one of three useful fashions.

Note Almost all portions of the report have 132 columns. Be advised that before printing, your printers should be set for compressed print. Before generating your reports, please review the next page and familiarize yourself with the map of windows that will guide your report generation.



1 Diagram continued on next page.



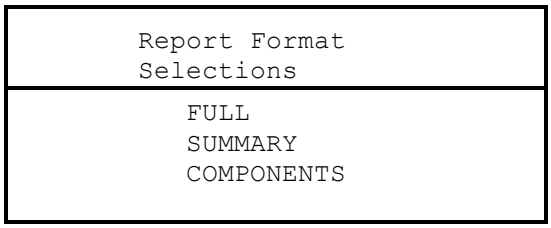
DBAnalyzer REPORT DESCRIPTION

Window Number	Description
10	Report Format Selections
20	Report Components
30	Domain Sort Order
40	Select Specific STORAGE AREAS
60	Select Specific VIEWS
70	Specify either ALL or SELECTED
80	Specify either COLUMNS or NO COLUMNS
90	Select Specific TABLES
100	Select Execution Options, ONLINE or BATCH
105	If in batch, print report when completed
110	Select Report Review Options
120	Select from Available Printer Queues

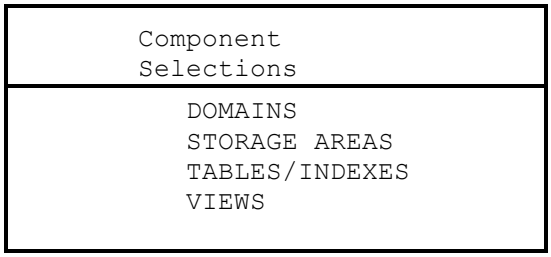
See the **DBAnalyzer Help** section on page 73 for more information on the various output options.

All of the above information is available via the DBAnalyzer report. This report can be accessed when the **F20** key is pressed or the **OUTPUT** menu option is selected. When the **OUTPUT** option is chosen, you will be prompted to enter a report file name. The default is **DBANALYZR.LST**. By default, the report file will be located in your current default directory.

After entering the filename, you will be presented with a window of options used to determine the format of the report.

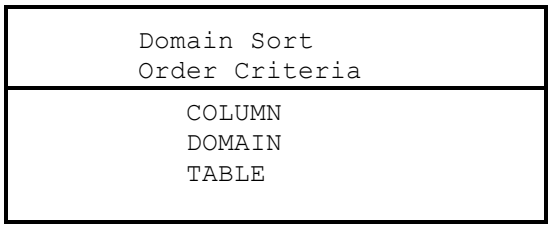


If you select either **FULL** or **SUMMARY**, you will be prompted to choose whether to run the report online or in batch. See **Execution Options** on page 39 for more information. If you select **COMPONENTS**, you will be presented with a window of items to be included on the report.

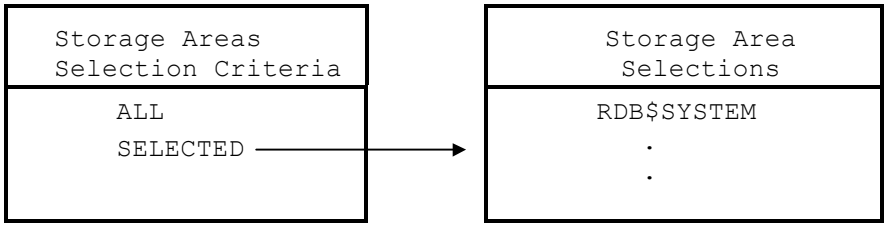


Use the **UP** and **DOWN** arrow keys to move through the window(s). Press **Select** to highlight an option to be included on the report. More than one component can be selected. Press **Return** when the selection process is completed.

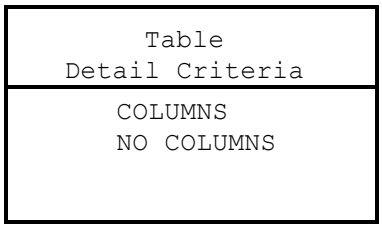
If you elect to include **DOMAINS** in the report listing, DBAnalyzer will present a window to determine the sort order for domains on the final report. Arrow between the three options and press **Return** for the one desired.



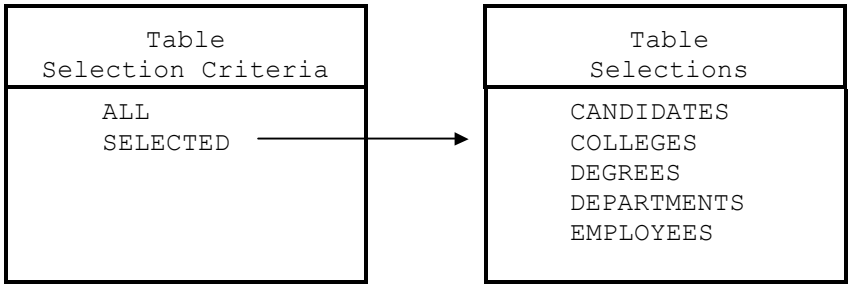
If you elect to include **STORAGE AREAS** in the report listing, then DBAnalyzer will present a window enabling you to choose whether to include ALL storage areas or only SELECTED areas. If you indicate **Selected Areas**, then a window will be presented to indicate which storage areas to include in the report. You can press **Select** for one or more storage areas on which to report. Pressing **Return** will complete the selection process.



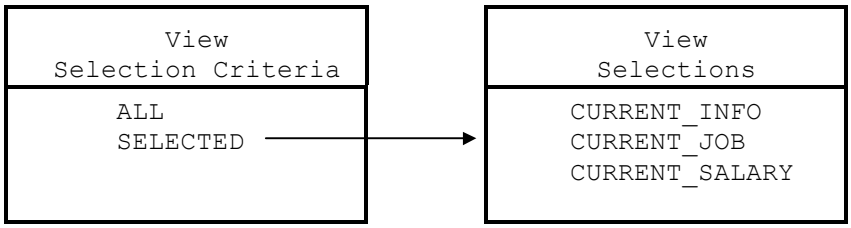
If you elected to include TABLES/INDEXES in the report, DBAnalyzer will present a window to select the level of detail for the **TABLE** report.



You will then be asked to select the tables on which to report. You may specify either **ALL** tables or **SELECTED**. If the **SELECTED** option is chosen, a window listing all the tables in the database will be presented. You can then arrow through the tables, highlighting the ones to be reported with the **Select** key. Pressing **Return** will complete the selection process.

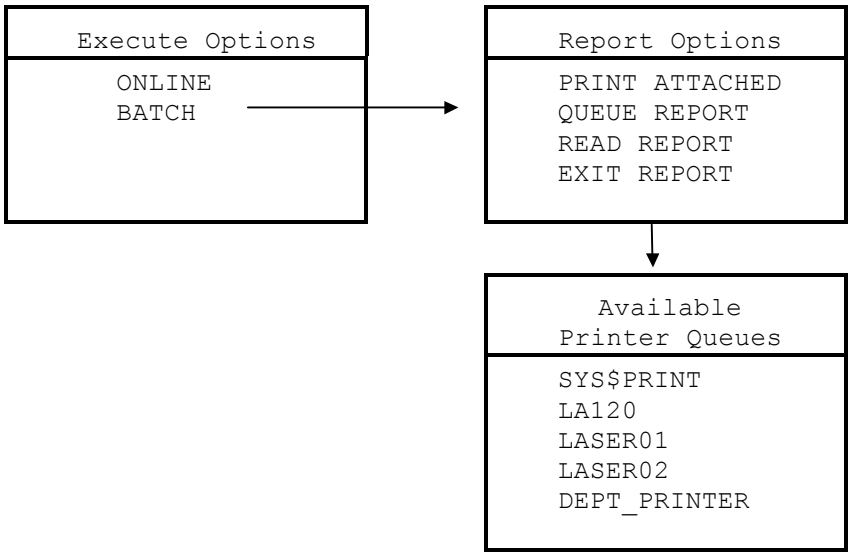


If you elect to include views in the report, you will be asked to select the views for reporting. You may specify either **ALL** views or **SELECTED**. If the **SELECTED** option is chosen, a window with all of the views will be presented. You can arrow through the views, highlighting the ones to be reported. Pressing **Return** will complete the selection process.



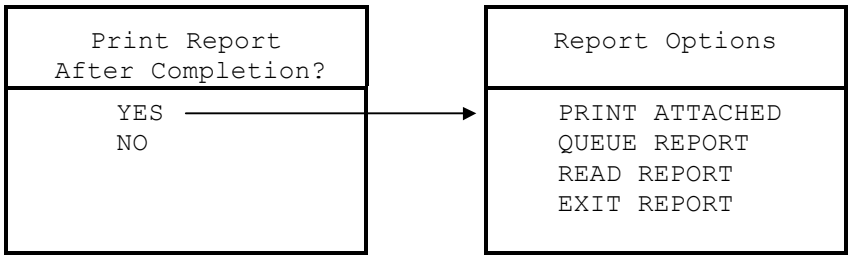
Execution Options:

Once the report specification is completed, you will be asked if the report is to be run **ONLINE** or in **BATCH**. A window will be used to present you with this option. If executed online, you will be presented with a window to select an available printer queue when the report completes.



Only one option can be selected at any one time from this menu. As long as you are selecting report options, this menu will continue to be presented.

If you elect to execute the report in batch, you will be asked whether or not to automatically print the report upon completion. If yes, you will be prompted for a queue to be used for the output. The queues are selected from among the print queues available on the system.



See the section on **BATCH Use of DBAnalyzer** on page 17 for information on creating the DBAnalyzer report in BATCH.

Sample DBAnalyzer Windows

MACRO MODE 1: (Tables with the Highest Record Counts)

```
        Tables with the Highest Record Counts
                   Macro Window 1


SALARY_HISTORY                          729
JOB_HISTORY                             274
DEGREES                                 165
EMPLOYEES                               100
DEPARTMENTS                              26
```

MACRO MODE 2: (Indexes with the Most Duplicates)

```
           Indexes with the Most Duplicates
                    Macro Window 2


DEG_COLLEGE_CODE                         12
SH_EMPLOYEE_ID                            7
JH_EMPLOYEE_ID                            2
DEG_EMP_ID                                1
EMP_LAST_NAME                             1
```

MACRO MODE 3: (Tables with the Most Columns)

```
         Tables with the Most Columns
                Macro Window 3


EMPLOYEES                               12
JOB_HISTORY                              6
COLLEGES                                 5
DEGREES                                  5
DEPARTMENTS                              5
```

MACRO MODE 4: (Tables with the Largest Record Size)

```
        Tables with the Largest Record Size
                  Macro Window 4


CANDIDATES                              280
EMPLOYEES                               112
COLLEGES                                 56
DEPARTMENTS                              47
JOB_HISTORY                              34
```

MACRO MODE 5: (SORTED Indexes Recommended to be HASHED)

```
SORTED Indexes Recommended to be HASHED
                  Macro Window 5


Index / Table
DEG_EMP_ID / DEGREES
EMP_EMPLOYEE_ID / EMPLOYEES
JH_EMPLOYEE_ID / JOB_HISTORY
SH_EMPLOYEE_ID / SALARY_HISTORY
```

MACRO MODE 6: (HASHED Indexes Recommended to be SORTED)

```
HASHED Indexes Recommended to be SORTED
                  Macro Window 6


              *** None were found ***
```

MACRO MODE 7: (Tables with the Most Indexes)

```
          Tables with the Most Indexes
                Macro Window 7


INVOICE_INFORMATION                    5
EXPENSES                               3
INVOICE_LINE_ITEMS                     3
MEAL_EXPENSES                          3
TIME_WEEK_HEADER                       3
```

MACRO MODE 8: (Non-Indexed Tables with the Most Records)

```
      Non-Indexed Tables with the Most Records
                Macro Window 8


             *** None were found ***
```

MACRO MODE 9: (Storage Areas with the Most Mapped Items)

```
      Storage Areas with the Most Mapped Items
                Macro Window 9


CONTRACT_INDEX_C1                      4
PRODUCTS_INDEX_C2                      4
CONTRACT_ITEMS_C3                      3
CONTRACT_AREA                          2
CONTRACT_ITEMS_C5                      2
```

MACRO MODE 10: (Storage Areas with the Most File Extensions)

```
      Storage Areas with the Most File Extensions
                  Macro Window 10


        TIME_TABLE                              8
        RDB$SYSTEM                              4
        TIME_INDEX                              2
        CONTRACT_INDEX_C1                       1
        CONTRACT_INDEX_C2                       1
```

MACRO MODE 11: (RDA Files with the Most Extension Blocks)

```
      RDA Files with the Most Extension Blocks
                  Macro Window 11


        RDB$SYSTEM                          3,608
        TR_TABLE                            3,396
        PRODUCT_INDEX_P5                      672
        TIME_INDEX                            597
        PAYROLL_INDEX                         576
```

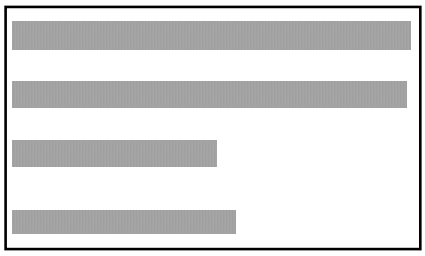
MACRO MODE 12: (SNP Files with the Most Extension Blocks)

```
      SNP Files with the Most Extension Blocks
                  Macro Window 12


        CONTRACT_INDEX_C6                     384
        INVOICE_INDEX_i1                      384
        PAYROLL_INDEX_R5                      384
        TIME_TABLE                            384
        TIME_ITEM_TABLE                       384
```

MACRO MODE 13: (Database Storage Area Extension Summary)

```
        Database Storage Area Extension Summary
                    Macro Window 13


        Total RDA Extensions:                  59
        Initial RDA Alloc (blocks)          5,635
        Current RDA Alloc (blocks)         29,389
        Initial SNP Alloc (blocks)            656
        Current SNP Alloc (blocks)          4,952
```

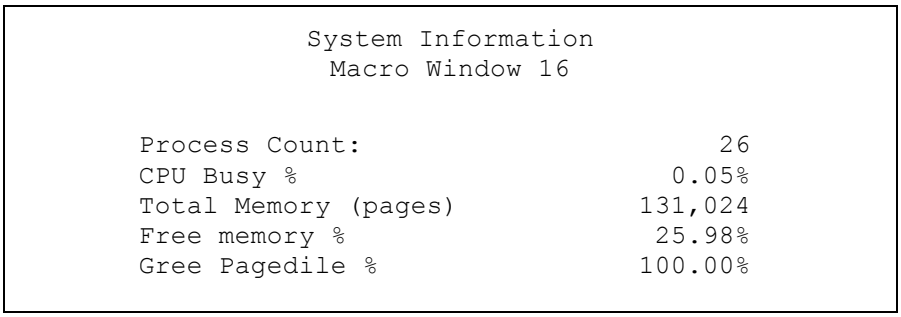
MACRO MODE 14: (Database Integrity Ratings)



MACRO MODE 15: (Database Recovery Summary)

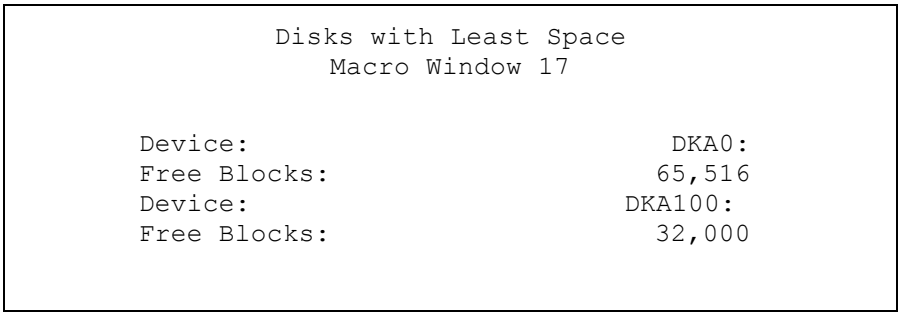
```
              Database Recovery Summary
                   Macro Window 15


        AIJ Enabled:                             N
        Last Db Backup:                      Never
        Recovery Buffers:                       20
        Buffer Size                       6 blocks
```

MACRO MODE 16: (System Information)



```
                System Information
                  Macro Window 16


      Process Count:                          26
      CPU Busy %                           0.05%
      Total Memory (pages)               131,024
      Free memory %                       25.98%
      Gree Pagedile %                    100.00%
```

MACRO MODE 17: (Disks with Least Space)



```
              Disks with Least Space
                  Macro Window 17


      Device:                               DKA0:
      Free Blocks:                        65,516
      Device:                            DKA100:
      Free Blocks:                        32,000
```

MICRO-TABLE MODE

```
        MICRO Mode:  Rdb Tables/Views


    Table:  DEGREES
    Cardinality (Record Count):        165
    Number of Columns:                   5
    Number of Bytes (Record Size)       29
     Storage Area:      RDB$SYSTEM
    Number of Indexes:          2
    No Unique Indexes Exist for This Table
```

MICRO-INDEX MODE

```
          MICRO Mode:  Rdb Indexes


    Table:          DEGREES
    Index (002):DEG_COLLEGE_CODE
    Index Type   :  SORTED, NON-UNIQUE
    Avg Dups     :           12
     Storage Area:      RDB$SYSTEM
    Index columns:          2
    1st column:   COLLEGE_CODE
```

MICRO-STORAGE AREA MODE

```
          MICRO Mode:  Rdb Storage Areas


      Storage Area: COMP_NAME_INDEX_AREA
      Page Size:    4 blocks,  Format: UNIFORM
      Number of Extents    :               1
      RDA Extensions (blocks):           194
      SNP Extensions (blocks):       0
      ------------Stored Elements------------
      Tables:   0   Sorted:   1   Hashed:   0
```

Sample DBAnalyzer Report

```
************************************************************************
*
*  Filename: ALI_DEFAULT_DIR:DBANALYZR.LST
*
*  Output generated by DBAnalyzer V6.0
*  03/29/2006 15:29:59
*
*  Database: MF_PERSONNEL (Rdb V7.2-010)
*
************************************************************************

                        === Rdb Analysis Summary ===

                   Complexity Rating: 8 (Low)
                   Integrity Rating: 150.00 (High)
                   Tune Rating: 32 (Very little tuning needed)
                   Indices to Review: 50%

Database Structure -
  Number of Tables       : 10          Number of Tbl Records: 1,333
  Avg Recs per Table     : 133         Number of Views      : 3
  Number of Indices      : 10          Hashed Index %       : 20%
  Number of Table Columns: 51          Number of Domains    : 28
  Number of Constraints  : 15          Number of Triggers   : 3

Open Characteristics -
  Open Mode              : Automatic   Maximum Users        : 50
  Current Users          : 1           Number of Nodes      : 16
  Open on other nodes    : NO

Database Recovery -
  Last Db Backup: NEVER                AIJ Enabled          : N
  Recovery Buffers       : 20

Database Buffers -
  Global Buffers         : N           Buffer Size          : 6 blocks
  # of Local Buffers     : 20

Database Locking -
  Adjustable Lock Gran   : Y           Lock Timeout Interval: 0

Database Storage -
  Non-System Areas       : 9           RDA Ext. Block %     : 68%
  SNP Ext. Block %       : 97%
```

```
DBAnalyzer                                                                                                    Page 2
Rdb Name: MF_PERSONNEL

Rdb Analysis Summary (continued)...
================================================================================================================

System Information -
  Process Count          : 26          CPU Count            : 1
  CPU Busy %             : 0.05%       Total Memory         : 131024 pages
  Free Memory %          : 25.98%      Free Pagefile %      : 100.00%
  Free Swapfile %        : 100.00%

Disk Information -
  Device: QUARK$DKA0:
  Free Blocks            : 62,516,608  Avg Queue Length     : 0.0

  Device: QUARK$DKA100:
  Free Blocks            : 68,858,720  Avg Queue Length     : 0.0
```

```
DBAnalyzer                                                                                                    Page 3
Rdb Name: MF_PERSONNEL

Rdb Analysis Summary (continued)...
================================================================================================================================


                        Storage Area Page Size Distribution
                              Frequency   Page Size
                              ---------   ---------
                                  9    ...    2
                                  1    ...    6
                              ---------
                                 10

              Cardinality Distribution for Tables and Their Indices
          Cardinality Range          #Tables   #Indices   Total   Accum
    ---------------------------     -------   --------   -----   -----
    1,000,000 or more records:          0          0       0       0
    100,000 to 999,999        :         0          0       0       0
    50,000 to 99,999          :         0          0       0       0
    10,000 to 49,999          :         0          0       0       0
    5,000 to 9,999            :         0          0       0       0
    2,500 to 4,999            :         0          0       0       0
    1,000 to 2,499            :         0          0       0       0
    500 to 999                :         1          1       2       2
    250 to 499                :         1          2       3       5
    100 to 249                :         2          5       7      12
    0 to 99                   :         6          2       8      20
```

```
DBAnalyzer                                                                                                    Page 4
Rdb Name: MF_PERSONNEL

Rdb Analysis (continued)...
================================================================================================================================
Tables with the Highest Record Counts   Indices with the Most Duplicates
-------------------------------------   -------------------------------------
SALARY_HISTORY                    729   DEG_COLLEGE_CODE                   12
JOB_HISTORY                       274   SH_EMPLOYEE_ID                      7
DEGREES                           165   JH_EMPLOYEE_ID                      2
EMPLOYEES                         100   JOB_HISTORY_HASH                    2
DEPARTMENTS                        26   DEG_EMP_ID                          1

Tables with the Most Columns            Tables with the Largest Record Size
-------------------------------------   -------------------------------------
EMPLOYEES                         12    CANDIDATES                        280
JOB_HISTORY                        6    EMPLOYEES                         112
COLLEGES                           5    COLLEGES                           56
DEGREES                            5    DEPARTMENTS                        47
DEPARTMENTS                        5    JOB_HISTORY                        34

SORTED Indices Recommend to be HASHED   HASHED Indices Recommend to be SORTED
-------------------------------------   -------------------------------------
Index / Table...                                *** None were found ***
DEG_EMP_ID / DEGREES
DEG_COLLEGE_CODE / DEGREES
EMP_EMPLOYEE_ID / EMPLOYEES
JH_EMPLOYEE_ID / JOB_HISTORY
SH_EMPLOYEE_ID / SALARY_HISTORY

Tables with the most Indices            Non-Indexed Tables with the Most Records
-------------------------------------   ---------------------------------------
EMPLOYEES                          3    JOBS                               15
DEGREES                            2    CANDIDATES                          3
JOB_HISTORY                        2    RESUMES                             3
COLLEGES                           1    WORK_STATUS                         3
DEPARTMENTS                        1
```

```
DBAnalyzer                                                                                                    Page 5
Rdb Name: MF_PERSONNEL

Rdb Analysis (continued)...
================================================================================================================================
Storage Areas with Most Mapped Items    Storage Areas with Most File Extensions
------------------------------------    ---------------------------------------
RDB$SYSTEM                           8  RDB$SYSTEM                            8
EMPIDS_LOW                           4  SALARY_HISTORY                        1
EMPIDS_MID                           4
EMPIDS_OVER                          4
EMP_INFO                             3

RDA Files with Most Extension Blocks    SNP Files with Most Extension Blocks
------------------------------------    ------------------------------------
RDB$SYSTEM                       1,868  DEPARTMENTS                      7,470
SALARY_HISTORY                     200  RDB$SYSTEM                         792
                                        EMPIDS_LOW                         198
                                        EMPIDS_MID                         198
                                        EMPIDS_OVER                        198

Storage Area Extension Summary
--------------------------------------
Total RDA Extensions :               9
Init RDA Alloc (blks):             972
Curr RDA Alloc (blks):           3,060
Init SNP Alloc (blks):             320
Curr SNP Alloc (blks):           9,176
```

```
DBAnalyzer                                                                    Page 6
Rdb Name: MF_PERSONNEL

Rdb Analysis (continued)...
================================================================================================================================

Database Complexity
-------------------
    The complexity rating is a weighted measure of the database
design and its stored records.  A rating of 8 indicates a
relatively small database.  Tuning requirements are simple.
    The largest factor in this rating is the domain count.
It accounts for 13% of the complexity rating.  The next largest
component of this rating is the column count, which is also 13%.
The complexity will increase as records and Rdb items (e.g.
tables, columns) are added.

Database Tune Rating
--------------------
    The tune rating is a composite measure of the physical storage
design as it applies to the logical structure of this database.
The complexity rating of 8 and the tune rating of 32 indicate
that the physical storage strategy should allow this database to
perform reasonably well and allow for some growth in complexity
without noticeable performance degradation.
    The tune rating measures significant factors that affect the
physical storage design.  It does not consider every factor, but
does objectively measure factors critical to successful Rdb tuning.
Remember that overall performance is a function of many things,
including system load, system tuning, and application design, in
addition to the physical storage strategy.
    DBTune, a complementary product, can be used to improve
the tune rating for a database.  DBTune uses volume, environment,
and activity data to tune storage areas, indices, and database
parameters.  SQL scripts to implement these tuning changes are
automatically generated as well as reports which give advice
on tuning options and SYSUAF/SYSGEN parameter settings.

Database Storage Area Allocation
```

```
DBAnalyzer                                                                      Page 7
Rdb Name: MF_PERSONNEL

Rdb Analysis (continued)...
================================================================================================================================
-------------------------------   The 'Storage Area Allocation' graphs indicate the percentage of
allocated space that has been extended for both RDA and SNP files.
Of the total RDA pages, a large percentage (68%) have been
extended as these storage areas have been loaded.   RDA pages are
extended when the data requirements for tables and/or indices exceed
the existing allocation of pages.  The RDA areas have been extended
9 times.

    Of the total SNP pages, a large percentage (97%) have been
created as these storage areas have been utilized.  SNP files are
used to enable READ-only transactions to access data concurrently
while WRITE transactions are active.  SNP pages are only used if
SNAPSHOTS ARE ENABLED.

Database Index Analysis
-----------------------
    The index analysis graphs show that 20% of the database indices are HASHED.
Thus, 80% of the indices are SORTED.  DBAnalyzer reviewed the indices
and found that five of the SORTED indices are candidates to be HASHED
and none of the HASHED indices are candidates to be SORTED.  Review
MACRO windows 5 and 6 to see which indices have been selected.
    DBAnalyzer looks for certain key words within the index columns.  It
assumes certain types of queries will be made based on these key words.
The person responsible for maintaining the database should review
actual usage to determine whether to modify the index.
    NOTE:  HASHED indices facilitate exact match queries.
           They incur narrow locks for updates.
           SORTED indices facilitate sequential retrievals.
           They incur broader lock contention for updates
           than HASHED indices.

MACRO and MICRO windows
-----------------------
```

```
DBAnalyzer                                                                                    Page 8
Rdb Name: MF_PERSONNEL

Rdb Analysis (continued)...
=============================================================================================================
    DBAnalyzer is a tool to increase the productivity of theindividual who manages Rdb databases.  The overview
provides
database-wide objective measurements.  The MACRO windows
highlight significant items that affect the database, its
applications, and its users.  The MICRO windows provide
additional details to assist in making decisions to maintain
and improve the usefulness of an Rdb database.


Database Integrity Ratings
--------------------------
    The Database Integrity Rating is an overall measure of the utilization
 of database constraints.  A rating of 150.00 indicates that few additional
constraints could be implemented to improve database integrity.  As
more tables and indices are added to the database, this rating may
drop unless a corresponding number of constraints are added as well.
Following are ratings that focus on more specific areas of database
integrity.

    The Columnar Integrity Rating considers column-level constraints such
as CHECK, NOT NULL, and FOREIGN KEY constraints that enforce data validation
in and between tables.  A rating of 45.00 indicates that additional
constraints could be implemented to improve the columnar integrity of
the database.  An example of such a column constraint is:
     CHECK(STATUS IN ('ACTIVE', 'INACTIVE') OR STATUS IS NULL).

    The Referential Integrity Rating considers UNIQUE, PRIMARY KEY,
and FOREIGN KEY constraints that enforce uniqueness or referential
validation in and between tables.  A rating of 33.33 indicates
that additional foreign key constraints could be implemented to
improve the referential integrity of the database.

    The Referential Efficiency Rating measures the extent to which existing
indices mirror UNIQUE and PRIMARY KEY constraints.  If such a constraint
```

```
DBAnalyzer                                                                    Page 9
Rdb Name: MF_PERSONNEL

Rdb Analysis (continued)...
================================================================================================================
has a matching unique index, the Rdb optimizer can evaluate theconstraint more efficiently by taking advantage of the
underlying
index.  A rating of 66.67 indicates that additional unique
indices could be implemented to improve the referential efficiency
of the database.
```

```
DBAnalyzer                                                                    Page 10
Rdb Name: MF_PERSONNEL

Storage Area Information:
=============================================================================================================
Area: DEPARTMENTS                  (PgSz: 2, Fmt:MIX Tbls:1, Sort:1, Hash:0, Exts/blks--RDA: 0/0, SNP: 16/7470)
            Stored              Elmnt     Cardin-    Column    Byte     Index            Average
           Elements             Type       ality     Count     Count    Type     Unique   Dups         Comments
------------------------------  -----   ----------  ------   ------    ------   ------  -------  --------------------
----------------
DEPARTMENTS                     TABLE   26               5       47                              PVI:
DEPARTMENTS_INDEX
DEPARTMENTS_INDEX               INDEX                    1        4    SORTED   YES        N/A   TBL: DEPARTMENTS

Area: EMPIDS_LOW                   (PgSz: 2, Fmt:MIX Tbls:2, Sort:0, Hash:2, Exts/blks--RDA: 0/0, SNP: 1/198)
            Stored              Elmnt     Cardin-    Column    Byte     Index            Average
           Elements             Type       ality     Count     Count    Type     Unique   Dups         Comments
------------------------------  -----   ----------  ------   ------    ------   ------  -------  --------------------
----------------
EMPLOYEES                       TABLE   100             12      112                              PVI: EMPLOYEES_HASH
JOB_HISTORY                     TABLE   274              6       34                              PVI:
JOB_HISTORY_HASH
EMPLOYEES_HASH                  INDEX                    1        5    HASHED   YES        N/A   TBL: EMPLOYEES
JOB_HISTORY_HASH                INDEX                    1        5    HASHED   NO           2   TBL: JOB_HISTORY

Area: EMPIDS_MID                   (PgSz: 2, Fmt:MIX Tbls:2, Sort:0, Hash:2, Exts/blks--RDA: 0/0, SNP: 1/198)
            Stored              Elmnt     Cardin-    Column    Byte     Index            Average
           Elements             Type       ality     Count     Count    Type     Unique   Dups         Comments
------------------------------  -----   ----------  ------   ------    ------   ------  -------  --------------------
----------------
EMPLOYEES                       TABLE   100             12      112                              PVI: EMPLOYEES_HASH
JOB_HISTORY                     TABLE   274              6       34                              PVI:
JOB_HISTORY_HASH
EMPLOYEES_HASH                  INDEX                    1        5    HASHED   YES        N/A   TBL: EMPLOYEES
JOB_HISTORY_HASH                INDEX                    1        5    HASHED   NO           2   TBL: JOB_HISTORY

Area: EMPIDS_OVER                  (PgSz: 2, Fmt:MIX Tbls:2, Sort:0, Hash:2, Exts/blks--RDA: 0/0, SNP: 1/198)
            Stored              Elmnt     Cardin-    Column    Byte     Index            Average
           Elements             Type       ality     Count     Count    Type     Unique   Dups         Comments
------------------------------  -----   ----------  ------   ------    ------   ------  -------  --------------------
----------------
EMPLOYEES                       TABLE   100             12      112                              PVI: EMPLOYEES_HASH
JOB_HISTORY                     TABLE   274              6       34                              PVI:
JOB_HISTORY_HASH
EMPLOYEES_HASH                  INDEX                    1        5    HASHED   YES        N/A   TBL: EMPLOYEES
JOB_HISTORY_HASH                INDEX                    1        5    HASHED   NO           2   TBL: JOB_HISTORY
```

```
DBAnalyzer                                                                    Page 11
Rdb Name: MF_PERSONNEL

Storage Area Information (continued)...
================================================================================================================
Area: EMP_INFO                        (PgSz: 2, Fmt:MIX Tbls:3, Sort:0, Hash:0, Exts/blks--RDA: 0/0, SNP: 0/0)
             Stored              Elmnt     Cardin-    Column    Byte     Index            Average
            Elements             Type       ality     Count     Count    Type     Unique   Dups         Comments
------------------------------   -----   ----------  ------   ------    ------   ------  -------  --------------------
----------------
COLLEGES                         TABLE   15               5       56
DEGREES                          TABLE   165              5       29
WORK_STATUS                      TABLE   3                3       23

Area: JOBS                            (PgSz: 2, Fmt:MIX Tbls:1, Sort:0, Hash:0, Exts/blks--RDA: 0/0, SNP: 0/0)
             Stored              Elmnt     Cardin-    Column    Byte     Index            Average
            Elements             Type       ality     Count     Count    Type     Unique   Dups         Comments
------------------------------   -----   ----------  ------   ------    ------   ------  -------  --------------------
----------------
JOBS                             TABLE   15               5       33

Area: RDB$SYSTEM                      (PgSz: 2, Fmt:UNI * LISTS * Tbls:1, Sort:7, Hash:0, Exts/blks--RDA:8/1868, SNP:
4/792)
             Stored              Elmnt     Cardin-    Column    Byte     Index            Average
            Elements             Type       ality     Count     Count    Type     Unique   Dups         Comments
------------------------------   -----   ----------  ------   ------    ------   ------  -------  --------------------
----------------
CANDIDATES                       TABLE   3                4      280
COLL_COLLEGE_CODE                INDEX                    1        4    SORTED   YES        N/A    TBL: COLLEGES
DEG_COLLEGE_CODE                 INDEX                    1        4    SORTED   NO          12    TBL: DEGREES
DEG_EMP_ID                       INDEX                    1        5    SORTED   NO           1    TBL: DEGREES
EMP_EMPLOYEE_ID                  INDEX                    1        5    SORTED   YES        N/A    TBL: EMPLOYEES
EMP_LAST_NAME                    INDEX                    1       14    SORTED   NO           1    TBL: EMPLOYEES
JH_EMPLOYEE_ID                   INDEX                    1        5    SORTED   NO           2    TBL: JOB_HISTORY
SH_EMPLOYEE_ID                   INDEX                    1        5    SORTED   NO           7    TBL: SALARY_HISTORY

Area: RESUMES                         (PgSz: 2, Fmt:MIX Tbls:1, Sort:0, Hash:0, Exts/blks--RDA: 0/0, SNP: 0/0)
             Stored              Elmnt     Cardin-    Column    Byte     Index            Average
            Elements             Type       ality     Count     Count    Type     Unique   Dups         Comments
------------------------------   -----   ----------  ------   ------    ------   ------  -------  --------------------
----------------
RESUMES                          TABLE   3                2       13
```

```
DBAnalyzer                                                                    Page 12
Rdb Name: MF_PERSONNEL

Storage Area Information (continued)...
================================================================================================================
Area: RESUME_LISTS                (PgSz: 6, Fmt:MIX * LISTS * Tbls:1, Sort:0, Hash:0, Exts/blks--RDA: 0/0, SNP: 0/0)
             Stored              Elmnt     Cardin-    Column    Byte     Index            Average
            Elements             Type       ality     Count     Count    Type     Unique   Dups         Comments
-------------------------------  -----   -----------  ------   ------    ------   ------  -------  --------------------
----------------

Area: SALARY_HISTORY              (PgSz: 2, Fmt:MIX Tbls:1, Sort:0, Hash:0, Exts/blks--RDA:1/200, SNP: 0/0)
             Stored              Elmnt     Cardin-    Column    Byte     Index            Average
            Elements             Type       ality     Count     Count    Type     Unique   Dups         Comments
-------------------------------  -----   -----------  ------   ------    ------   ------  -------  --------------------
----------------
SALARY_HISTORY                   TABLE   729              4       25
```

```
DBAnalyzer                                                                    Page 13
Rdb Name: MF_PERSONNEL

Individual Table Statistics...
=============================================================================================================
Table: CANDIDATES                      (Card: 3            Col: 4      Byt: 280     Idx: 0    Areas: RDB$SYSTEM)
            Columns for Table:
            CANDIDATES                            Column Domain             Domain/Datatype Description
            ------------------------------  ------------------------------  ---------------------------------
      1     CANDIDATE_STATUS                CANDIDATE_STATUS_DOM            VARCHAR(255)
      2     FIRST_NAME                      FIRST_NAME_DOM                  CHAR(10)
      3     LAST_NAME                       LAST_NAME_DOM                   CHAR(14)
      4     MIDDLE_INITIAL                  MIDDLE_INITIAL_DOM              CHAR(1)

                                                <<<<<>>>>>

Table: COLLEGES                        (Card: 15           Col: 5      Byt: 56      Idx: 1    Areas: EMP_INFO)
> Index: COLL_COLLEGE_CODE               (SORTED  Dup: UNIQUE  Col: 1    Byt: 4   Node: 430    Areas: RDB$SYSTEM)
            Columns for Table:
            COLLEGES                              Column Domain             Domain/Datatype Description
            ------------------------------  ------------------------------  ---------------------------------
      1     CITY                            CITY_DOM                        CHAR(20)
      2     COLLEGE_CODE                    COLLEGE_CODE_DOM                CHAR(4)
      3     COLLEGE_NAME                    COLLEGE_NAME_DOM                CHAR(25)
      4     POSTAL_CODE                     POSTAL_CODE_DOM                 CHAR(5)
      5     STATE                           STATE_DOM                       CHAR(2)

            Columns for Index:
            COLL_COLLEGE_CODE                     Column Domain             Domain/Datatype Description
            ------------------------------  ------------------------------  ---------------------------------
      1     COLLEGE_CODE                    COLLEGE_CODE_DOM                CHAR(4)

                                                <<<<<>>>>>

Table: DEGREES                         (Card: 165          Col: 5      Byt: 29      Idx: 2    Areas: EMP_INFO)
```

```
DBAnalyzer                                                                                                    Page 14
Rdb Name: MF_PERSONNEL

Individual Table Statistics (continued)...
=============================================================================================================================
> Index: DEG_COLLEGE_CODE                (SORTED  Dup: 12      Col: 1    Byt: 4   Node: deflt  Areas: RDB$SYSTEM)
> Index: DEG_EMP_ID                      (SORTED  Dup: 1       Col: 1    Byt: 5   Node: deflt  Areas: RDB$SYSTEM)
            Columns for Table:
            DEGREES                                  Column Domain              Domain/Datatype Description
            -------------------------------  -------------------------------   ---------------------------------
      1     COLLEGE_CODE                     COLLEGE_CODE_DOM                  CHAR(4)
      2     DEGREE                           DEGREE_DOM                        CHAR(3)
      3     DEGREE_FIELD                     DEGREE_FIELD_DOM                  CHAR(15)
      4     EMPLOYEE_ID                      ID_DOM                            CHAR(5)
      5     YEAR_GIVEN                       YEAR_DOM                          SMALLINT(0)

            Columns for Index:
            DEG_COLLEGE_CODE                         Column Domain              Domain/Datatype Description
            -------------------------------  -------------------------------   ---------------------------------
      1     COLLEGE_CODE                     COLLEGE_CODE_DOM                  CHAR(4)

            Columns for Index:
            DEG_EMP_ID                               Column Domain              Domain/Datatype Description
            -------------------------------  -------------------------------   ---------------------------------
      1     EMPLOYEE_ID                      ID_DOM                            CHAR(5)

                                                <<<<<>>>>>

Table: DEPARTMENTS                     (Card: 26            Col: 5      Byt: 47      Idx: 1    Areas: DEPARTMENTS)
> Index: DEPARTMENTS_INDEX               (SORTED  Dup: UNIQUE  Col: 1    Byt: 4   Node: 430    Areas: DEPARTMENTS)
            Columns for Table:
            DEPARTMENTS                              Column Domain              Domain/Datatype Description
            -------------------------------  -------------------------------   ---------------------------------
      1     BUDGET_ACTUAL                    BUDGET_DOM                        INTEGER(0)
      2     BUDGET_PROJECTED                 BUDGET_DOM                        INTEGER(0)
      3     DEPARTMENT_CODE                  DEPARTMENT_CODE_DOM               CHAR(4)
      4     DEPARTMENT_NAME                  DEPARTMENT_NAME_DOM               CHAR(30)
      5     MANAGER_ID                       ID_DOM                            CHAR(5)
```

```
DBAnalyzer                                                                                                    Page 15
Rdb Name: MF_PERSONNEL

Individual Table Statistics (continued)...
================================================================================================================================

            Columns for Index:
            DEPARTMENTS_INDEX                        Column Domain              Domain/Datatype Description
            -------------------------------  -------------------------------   ---------------------------------
      1     DEPARTMENT_CODE                  DEPARTMENT_CODE_DOM                CHAR(4)

                                                  <<<<<>>>>>

Table: EMPLOYEES                        (Card: 100           Col: 12     Byt: 112     Idx: 3    Areas: EMPIDS_LOW,
                                                                                                      EMPIDS_MID,
                                                                                                      EMPIDS_OVER)
> Index: EMPLOYEES_HASH                   (HASHED  Dup: UNIQUE  Col: 1    Byt: 5                Areas: EMPIDS_LOW,
                                                                                                      EMPIDS_MID,
                                                                                                      EMPIDS_OVER)
> Index: EMP_EMPLOYEE_ID                  (SORTED  Dup: UNIQUE  Col: 1    Byt: 5   Node: 430    Areas: RDB$SYSTEM)
> Index: EMP_LAST_NAME                    (SORTED  Dup: 1       Col: 1    Byt: 14  Node: deflt  Areas: RDB$SYSTEM)
            Columns for Table:
            EMPLOYEES                                Column Domain              Domain/Datatype Description
            -------------------------------  -------------------------------   ---------------------------------
      1     ADDRESS_DATA_1                   ADDRESS_DATA_1_DOM                 CHAR(25)
      2     ADDRESS_DATA_2                   ADDRESS_DATA_2_DOM                 CHAR(20)
      3     BIRTHDAY                         DATE_DOM                           DATE VMS
      4     CITY                             CITY_DOM                           CHAR(20)
      5     EMPLOYEE_ID                      ID_DOM                             CHAR(5)
      6     FIRST_NAME                       FIRST_NAME_DOM                     CHAR(10)
      7     LAST_NAME                        LAST_NAME_DOM                      CHAR(14)
      8     MIDDLE_INITIAL                   MIDDLE_INITIAL_DOM                 CHAR(1)
      9     POSTAL_CODE                      POSTAL_CODE_DOM                    CHAR(5)
      10    SEX                              SEX_DOM                            CHAR(1)
      11    STATE                            STATE_DOM                          CHAR(2)
      12    STATUS_CODE                      STATUS_CODE_DOM                    CHAR(1)

            Columns for Index:
            EMPLOYEES_HASH                           Column Domain              Domain/Datatype Description
            -------------------------------  -------------------------------   ---------------------------------
      1     EMPLOYEE_ID                      ID_DOM                             CHAR(5)
```

```
DBAnalyzer                                                                                                    Page 16
Rdb Name: MF_PERSONNEL

Individual Table Statistics (continued)...
================================================================================================================================

            Columns for Index:
            EMP_EMPLOYEE_ID                          Column Domain              Domain/Datatype Description
            -------------------------------  -------------------------------   ---------------------------------
      1     EMPLOYEE_ID                      ID_DOM                             CHAR(5)

            Columns for Index:
            EMP_LAST_NAME                            Column Domain              Domain/Datatype Description
            -------------------------------  -------------------------------   ---------------------------------
      1     LAST_NAME                        LAST_NAME_DOM                      CHAR(14)

                                                  <<<<<>>>>>

Table: JOBS                           (Card: 15            Col: 5      Byt: 33      Idx: 0    Areas: JOBS)
            Columns for Table:
            JOBS                                     Column Domain              Domain/Datatype Description
            -------------------------------  -------------------------------   ---------------------------------
      1     JOB_CODE                         JOB_CODE_DOM                       CHAR(4)
      2     JOB_TITLE                        JOB_TITLE_DOM                      CHAR(20)
      3     MAXIMUM_SALARY                   SALARY_DOM                         INTEGER(2)
      4     MINIMUM_SALARY                   SALARY_DOM                         INTEGER(2)
      5     WAGE_CLASS                       WAGE_CLASS_DOM                     CHAR(1)

                                                  <<<<<>>>>>

Table: JOB_HISTORY                    (Card: 274           Col: 6      Byt: 34      Idx: 2    Areas: EMPIDS_LOW,
                                                                                                    EMPIDS_MID,
                                                                                                    EMPIDS_OVER)
> Index: JH_EMPLOYEE_ID                 (SORTED  Dup: 2       Col: 1    Byt: 5   Node: deflt  Areas: RDB$SYSTEM)
> Index: JOB_HISTORY_HASH               (HASHED  Dup: 2       Col: 1    Byt: 5                Areas: EMPIDS_LOW,
                                                                                                    EMPIDS_MID,
                                                                                                    EMPIDS_OVER)
            Columns for Table:
            JOB_HISTORY                              Column Domain              Domain/Datatype Description
            -------------------------------  -------------------------------   ---------------------------------
      1     DEPARTMENT_CODE                  DEPARTMENT_CODE_DOM                CHAR(4)
      2     EMPLOYEE_ID                      ID_DOM                             CHAR(5)
```

```
DBAnalyzer                                                                                                    Page 17
Rdb Name: MF_PERSONNEL

Individual Table Statistics (continued)...
=================================================================================================================================
            Columns for Table:
            JOB_HISTORY                          Column Domain              Domain/Datatype Description
            -------------------------------  -------------------------------   --------------------------------
      3     JOB_CODE                         JOB_CODE_DOM                      CHAR(4)
      4     JOB_END                          DATE_DOM                          DATE VMS
      5     JOB_START                        DATE_DOM                          DATE VMS
      6     SUPERVISOR_ID                    ID_DOM                            CHAR(5)

            Columns for Index:
            JH_EMPLOYEE_ID                       Column Domain              Domain/Datatype Description
            -------------------------------  -------------------------------   --------------------------------
      1     EMPLOYEE_ID                      ID_DOM                            CHAR(5)

            Columns for Index:
            JOB_HISTORY_HASH                     Column Domain              Domain/Datatype Description
            -------------------------------  -------------------------------   --------------------------------
      1     EMPLOYEE_ID                      ID_DOM                            CHAR(5)

                                                  <<<<<>>>>>

Table: RESUMES                          (Card: 3             Col: 2      Byt: 13      Idx: 0    Areas: RESUMES)
            Columns for Table:
            RESUMES                              Column Domain              Domain/Datatype Description
            -------------------------------  -------------------------------   --------------------------------
      1     EMPLOYEE_ID                      ID_DOM                            CHAR(5)
      2     RESUME                           RESUME_DOM                        LIST OF BYTE VARYING(1)

                                                  <<<<<>>>>>

Table: SALARY_HISTORY                   (Card: 729           Col: 4      Byt: 25      Idx: 1    Areas: SALARY_HISTORY)
```

```
DBAnalyzer                                                                                                    Page 18
Rdb Name: MF_PERSONNEL

Individual Table Statistics (continued)...
================================================================================================================================
> Index: SH_EMPLOYEE_ID                  (SORTED  Dup: 7       Col: 1    Byt: 5   Node: deflt  Areas: RDB$SYSTEM)
            Columns for Table:
            SALARY_HISTORY                          Column Domain              Domain/Datatype Description
            ------------------------------  ------------------------------   ---------------------------------
      1     EMPLOYEE_ID                     ID_DOM                           CHAR(5)
      2     SALARY_AMOUNT                   SALARY_DOM                       INTEGER(2)
      3     SALARY_END                      DATE_DOM                         DATE VMS
      4     SALARY_START                    DATE_DOM                         DATE VMS

            Columns for Index:
            SH_EMPLOYEE_ID                          Column Domain              Domain/Datatype Description
            ------------------------------  ------------------------------   ---------------------------------
      1     EMPLOYEE_ID                     ID_DOM                           CHAR(5)

                                                <<<<<>>>>>

Table: WORK_STATUS                     (Card: 3             Col: 3      Byt: 23      Idx: 0    Areas: EMP_INFO)
            Columns for Table:
            WORK_STATUS                             Column Domain              Domain/Datatype Description
            ------------------------------  ------------------------------   ---------------------------------
      1     STATUS_CODE                     STATUS_CODE_DOM                  CHAR(1)
      2     STATUS_NAME                     STATUS_NAME_DOM                  CHAR(8)
      3     STATUS_TYPE                     STATUS_DESC_DOM                  CHAR(14)
```

```
DBAnalyzer                                                                                                    Page 19
Rdb Name: MF_PERSONNEL

Individual View Statistics...
=============================================================================================================================

View: CURRENT_INFO
Columns                          Data Type   Domain
------------------------------   ---------   ------------------------------
LAST_NAME                             C       LAST_NAME_DOM
FIRST_NAME                            C       FIRST_NAME_DOM
ID                                    C       ID_DOM
DEPARTMENT                            C       DEPARTMENT_NAME_DOM
JOB                                   C       JOB_TITLE_DOM
JSTART                                DV      DATE_DOM
SSTART                                DV      DATE_DOM
SALARY                                I       SALARY_DOM
 SELECT
             CJ.LAST_NAME,
             CJ.FIRST_NAME,
             CJ.EMPLOYEE_ID,
              D.DEPARTMENT_NAME,
              J.JOB_TITLE,
             CJ.JOB_START,
             CS.SALARY_START,
             CS.SALARY_AMOUNT
        FROM CURRENT_JOB CJ,
             DEPARTMENTS D,
             JOBS J,
             CURRENT_SALARY CS
       WHERE CJ.DEPARTMENT_CODE = D.DEPARTMENT_CODE
         AND CJ.JOB_CODE = J.JOB_CODE
         AND CJ.EMPLOYEE_ID = CS.EMPLOYEE_ID;

View: CURRENT_JOB
Columns                          Data Type   Domain
------------------------------   ---------   ------------------------------
LAST_NAME                             C       LAST_NAME_DOM
FIRST_NAME                            C       FIRST_NAME_DOM
EMPLOYEE_ID                           C       ID_DOM
JOB_CODE                              C       JOB_CODE_DOM
DEPARTMENT_CODE                       C       DEPARTMENT_CODE_DOM
SUPERVISOR_ID                         C       ID_DOM
JOB_START                             DV      DATE_DOM
```

```
DBAnalyzer                                                                                                    Page 20
Rdb Name: MF_PERSONNEL

Individual View Statistics (continued)...
================================================================================================================================
   SELECT  E.LAST_NAME,
           E.FIRST_NAME,
           E.EMPLOYEE_ID,
          JH.JOB_CODE,
          JH.DEPARTMENT_CODE,
          JH.SUPERVISOR_ID,
          JH.JOB_START
     FROM JOB_HISTORY JH,
          EMPLOYEES E
    WHERE JH.EMPLOYEE_ID = E.EMPLOYEE_ID
          AND JH.JOB_END IS NULL;

View: CURRENT_SALARY
Columns                          Data Type   Domain
-------------------------------  ----------  -------------------------------
LAST_NAME                             C       LAST_NAME_DOM
FIRST_NAME                            C       FIRST_NAME_DOM
EMPLOYEE_ID                           C       ID_DOM
SALARY_START                          DV      DATE_DOM
SALARY_AMOUNT                         I       SALARY_DOM
   SELECT  E.LAST_NAME,
           E.FIRST_NAME,
           E.EMPLOYEE_ID,
          SH.SALARY_START,
          SH.SALARY_AMOUNT
     FROM SALARY_HISTORY SH,
          EMPLOYEES E
    WHERE SH.EMPLOYEE_ID = E.EMPLOYEE_ID
      AND SH.SALARY_END IS NULL;
```

```
DBAnalyzer                                                                    Page 21
Rdb Name: MF_PERSONNEL

Domain Summary...
================================================================================================================

          Domain Name              Column Name having Domain               Table/View Name          Datatype
Description
-------------------------------  -------------------------------  -------------------------------  --------------------
-----------

ADDRESS_DATA_1_DOM               ADDRESS_DATA_1                   EMPLOYEES                        CHAR(25)
 > standard definition for street addresses

ADDRESS_DATA_2_DOM               ADDRESS_DATA_2                   EMPLOYEES                        CHAR(20)
 > standard definition for apartments, suites, etc.

BUDGET_DOM                       BUDGET_ACTUAL                    DEPARTMENTS                      INTEGER(0)
 > standard definition for departmental budget
                                 BUDGET_PROJECTED                 DEPARTMENTS

CANDIDATE_STATUS_DOM             CANDIDATE_STATUS                 CANDIDATES                       VARCHAR(255)

CITY_DOM                         CITY                             COLLEGES                         CHAR(20)
 > standard definition of city or town
                                 CITY                             EMPLOYEES

COLLEGE_CODE_DOM                 COLLEGE_CODE                     COLLEGES                         CHAR(4)
 > standard definition of college code
                                 COLLEGE_CODE                     DEGREES

COLLEGE_NAME_DOM                 COLLEGE_NAME                     COLLEGES                         CHAR(25)
 > standard definition for college name

DATE_DOM                         BIRTHDAY                         EMPLOYEES                        DATE VMS
 > standard definition for complete dates
                                 JOB_END                          JOB_HISTORY
                                 JOB_START                        CURRENT_JOB
                                 JOB_START                        JOB_HISTORY
                                 JSTART                           CURRENT_INFO
                                 SALARY_END                       SALARY_HISTORY
                                 SALARY_START                     CURRENT_SALARY
                                 SALARY_START                     SALARY_HISTORY
                                 SSTART                           CURRENT_INFO

DEGREE_DOM                       DEGREE                           DEGREES                          CHAR(3)
 > standard definition for the kind of college degree awarded
```

```
DBAnalyzer                                                                                                    Page 22
Rdb Name: MF_PERSONNEL

Domain Summary (continued)...
================================================================================================================================

          Domain Name              Column Name having Domain               Table/View Name          Datatype
Description
-------------------------------  -------------------------------  -------------------------------  --------------------
-----------

DEGREE_FIELD_DOM                 DEGREE_FIELD                     DEGREES                          CHAR(15)
 > standard definition for description of college degree field

DEPARTMENT_CODE_DOM              DEPARTMENT_CODE                  CURRENT_JOB                      CHAR(4)
 > standard definition of department code
                                 DEPARTMENT_CODE                  DEPARTMENTS
                                 DEPARTMENT_CODE                  JOB_HISTORY

DEPARTMENT_NAME_DOM              DEPARTMENT                       CURRENT_INFO                     CHAR(30)
 > standard definition for department name
                                 DEPARTMENT_NAME                  DEPARTMENTS

FIRST_NAME_DOM                   FIRST_NAME                       CANDIDATES                       CHAR(10)
 > standard definition of first name
                                 FIRST_NAME                       CURRENT_INFO
                                 FIRST_NAME                       CURRENT_JOB
                                 FIRST_NAME                       CURRENT_SALARY
                                 FIRST_NAME                       EMPLOYEES

ID_DOM                           EMPLOYEE_ID                      CURRENT_JOB                      CHAR(5)
 > standard definition of employee id
                                 EMPLOYEE_ID                      CURRENT_SALARY
                                 EMPLOYEE_ID                      DEGREES
                                 EMPLOYEE_ID                      EMPLOYEES
                                 EMPLOYEE_ID                      JOB_HISTORY
                                 EMPLOYEE_ID                      RESUMES
                                 EMPLOYEE_ID                      SALARY_HISTORY
                                 ID                               CURRENT_INFO
                                 MANAGER_ID                       DEPARTMENTS
                                 SUPERVISOR_ID                    CURRENT_JOB
                                 SUPERVISOR_ID                    JOB_HISTORY

JOB_CODE_DOM                     JOB_CODE                         CURRENT_JOB                      CHAR(4)
 > standard definition of job code
```

```
DBAnalyzer                                                                    Page 23
Rdb Name: MF_PERSONNEL

Domain Summary (continued)...
================================================================================================================

          Domain Name              Column Name having Domain               Table/View Name          Datatype
Description
-------------------------------  -------------------------------  -------------------------------  --------------------
-----------
                                 JOB_CODE                         JOBS
                                 JOB_CODE                         JOB_HISTORY

JOB_TITLE_DOM                    JOB                              CURRENT_INFO                     CHAR(20)
 > standard definition for job title
                                 JOB_TITLE                        JOBS

LAST_NAME_DOM                    LAST_NAME                        CANDIDATES                       CHAR(14)
 > standard definition of last name
                                 LAST_NAME                        CURRENT_INFO
                                 LAST_NAME                        CURRENT_JOB
                                 LAST_NAME                        CURRENT_SALARY
                                 LAST_NAME                        EMPLOYEES

MIDDLE_INITIAL_DOM               MIDDLE_INITIAL                   CANDIDATES                       CHAR(1)
 > standard definition of middle initial
                                 MIDDLE_INITIAL                   EMPLOYEES

POSTAL_CODE_DOM                  POSTAL_CODE                      COLLEGES                         CHAR(5)
 > standard definition of ZIP
                                 POSTAL_CODE                      EMPLOYEES

RESUME_DOM                       RESUME                           RESUMES                          LIST OF BYTE
VARYING(1)

SALARY_DOM                       MAXIMUM_SALARY                   JOBS                             INTEGER(2)
 > standard definition of salary
                                 MINIMUM_SALARY                   JOBS
                                 SALARY                           CURRENT_INFO
                                 SALARY_AMOUNT                    CURRENT_SALARY
                                 SALARY_AMOUNT                    SALARY_HISTORY

SEX_DOM                          SEX                              EMPLOYEES                        CHAR(1)
 > standard definition for sex

STATE_DOM                        STATE                            COLLEGES                         CHAR(2)
 > standard definition of state
```

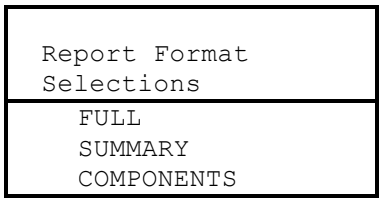
```
DBAnalyzer                                                                                                    Page 24
Rdb Name: MF_PERSONNEL

Domain Summary (continued)...
================================================================================================================================

          Domain Name              Column Name having Domain               Table/View Name          Datatype
Description
-------------------------------  -------------------------------  -------------------------------  --------------------
-----------
                                 STATE                            EMPLOYEES

STATUS_CODE_DOM                  STATUS_CODE                      EMPLOYEES                        CHAR(1)
 > standard definition of employment status codes
                                 STATUS_CODE                      WORK_STATUS

STATUS_DESC_DOM                  STATUS_TYPE                      WORK_STATUS                      CHAR(14)
 > standard definition for full-time/part-time description

STATUS_NAME_DOM                  STATUS_NAME                      WORK_STATUS                      CHAR(8)
 > standard definition for active/inactive description

WAGE_CLASS_DOM                   WAGE_CLASS                       JOBS                             CHAR(1)
 > standard definition for wage classification

YEAR_DOM                         YEAR_GIVEN                       DEGREES                          SMALLINT(0)
 > standard definition for year-only date values
*
* Parameters used to generate the above DBAnalyzer report:
* ---------------------------------------------------------
*    dba_report_name = DBANALYZR.LST
*    dba_select_full = YES
*    dba_select_brief = NO
*    dba_select_narrative = YES
*    dba_select_storage_areas = YES
*    dba_select_tables = YES
*    dba_table_columns = YES
*    dba_table_indices = YES
*    dba_table_indices_option = FULL
*
*    dba_select_views = YES
*    dba_view_option = FULL
*
*    dba_select_domains = YES
*    dba_domain_sort_order = DOMAIN
*    dba_domain_print_keyfields = NO
*
*    dba_print_report = NO
*    dba_printer =
*    dba_printer_options =
```

DBAnalyzer Help

SET_REPORT_NAME

Changing the report output name from the default DBANALYZR.LST offers you an opportunity to give the report a more meaningful name that is related to the contents. If multiple users are running DBAnalyzer and are generating reports on or about the same time, it is advisable to change the report name.

REPORT_PARAMETERS

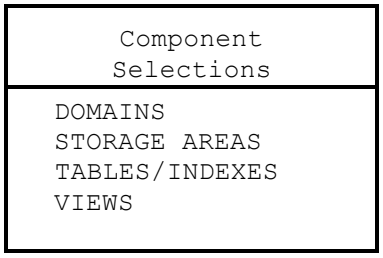


There are two basic formats available for the report(s) generated by DBAnalyzer. The **FULL** option will generate a comprehensive report about all aspects of the database.

The **SUMMARY** report only reports the information available in the MACRO windows of DBAnalyzer, plus a narrative explaining the complexity and tune rating of the database.

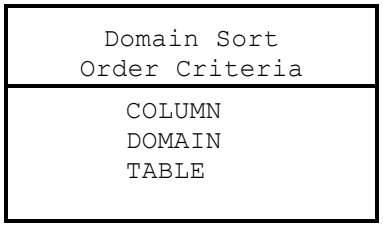
To tailor the report, you should select **COMPONENTS**. This will allow you to report on selected components of the database at various levels of detail. In this manner, you can report on selected storage areas or selected tables, as opposed to the entire database at one time.

REPORT_COMPONENTS



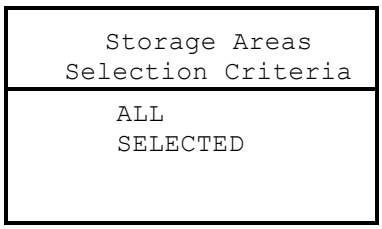
You can select from four options when formatting a report by components. Multiple selections are available from this menu. To report on a component, position the bar over an item using the arrow keys, and press the **Select** key. When the selection process is complete, press **Return**.

DOMAIN_SORT_ORDER



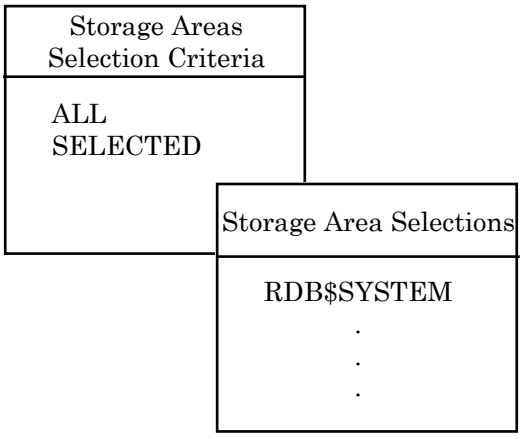
The **Domain Sort Order** determines the sorted order in which the domains and fields in the database are presented. In **DOMAIN** order, the fields will be sorted by the **DOMAIN** name. Thus, all fields sharing a common domain will be printed together. In **TABLE** order, all the fields in a given table will be printed together, yielding a table layout. In **FIELD** order, tables and domains are ordered by field name. This can be useful when looking for incorrect definitions of a field in different tables.

STORAGE_AREA_SELECTION



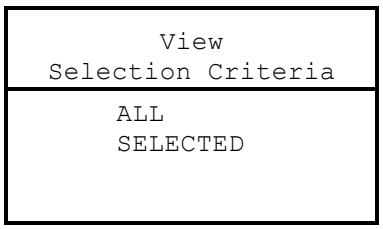
You can specify either **ALL** or **SELECTED** storage areas. If **ALL** is selected, then all storage areas in the database will be reported. If you opt for **SELECTED**, a window will be presented which will allow you to specify the area(s) to be reported.

SELECTED_STORAGE_AREAS



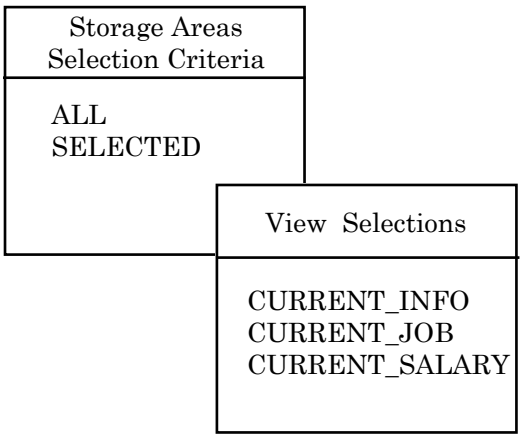
This window presents you with a selection of all storage areas in the database. You can select one or more storage areas by arrowing up and down through the window and highlighting the areas to be included by pressing the **Select** key. Pressing **Return** will include the highlighted areas in the final report.

VIEW_SELECTIONS



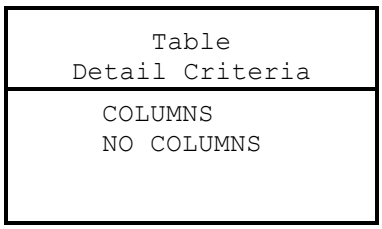
You can specify either **ALL** or **SELECTED** views. If **ALL** is selected, then all views in the database will be reported. If you opt for **SELECTED**, a window will be presented which will allow you to specify the view(s) to be reported.

SELECTED_VIEWS



This window presents you with a selection of all views in the database. You can select one or more views by arrowing up and down through the window and highlighting the ones to be included by pressing the **Select** key. Pressing **Return** will include the highlighted views in the final report.

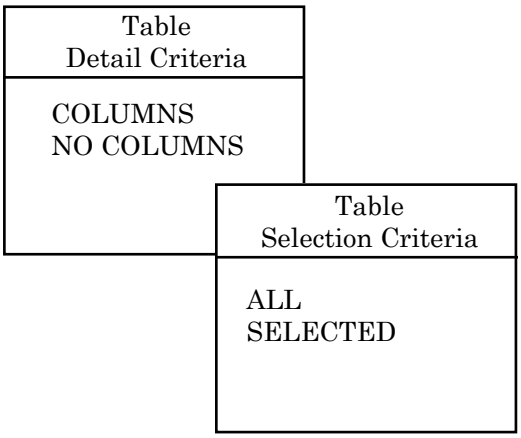
TABLE_DETAIL_OPTIONS



You can select from several levels of detail concerning the tables in the database. The **COLUMNS** option will give a listing of the columns in the table, columns in each index in the table, and such summary information as record size, column count, and cardinality.

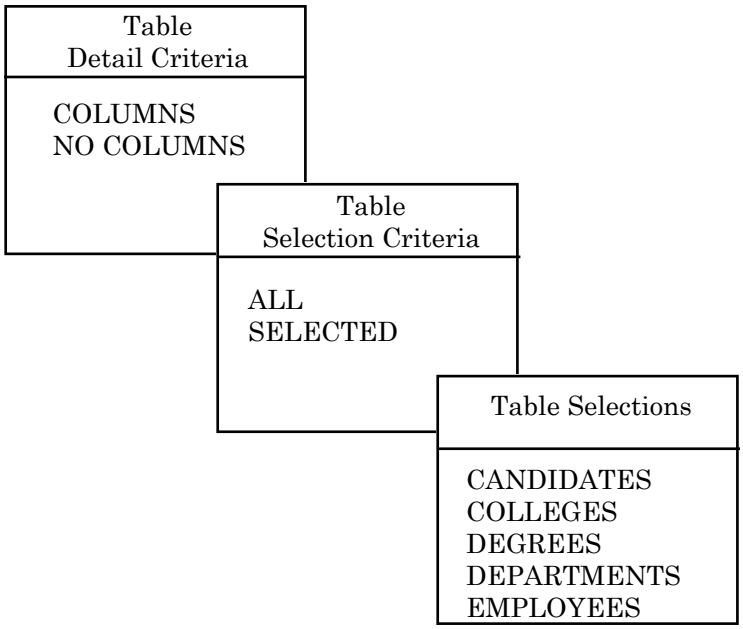
The **NO COLUMNS** option presents only summary information for tables/indexes and omits the column listings.

TABLE_SELECTIONS



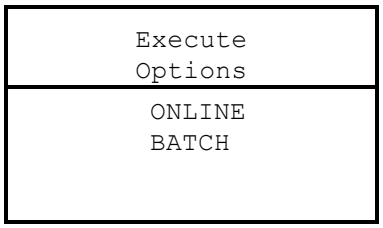
You can specify either **ALL** or **SELECTED** tables. If **ALL** is selected, then all tables in the database will be reported. If you opt for **SELECTED**, a window will be presented which will allow you to specify the table(s) to be reported.

SELECTED_TABLES



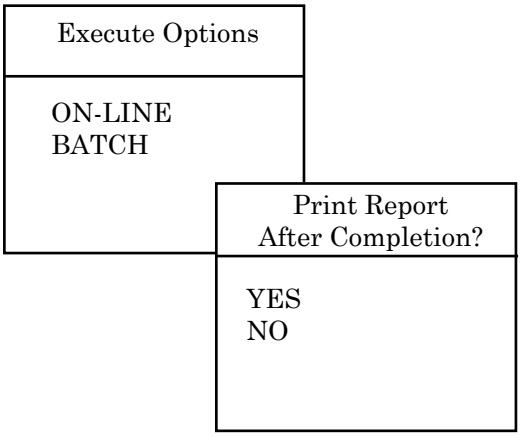
This window presents you with a selection of all tables in the database. You can select one or more tables by arrowing up and down through the window and highlighting the ones to be included by pressing the **Select** key. Pressing **Return** will include the highlighted tables in the final report.

EXECUTION_OPTIONS



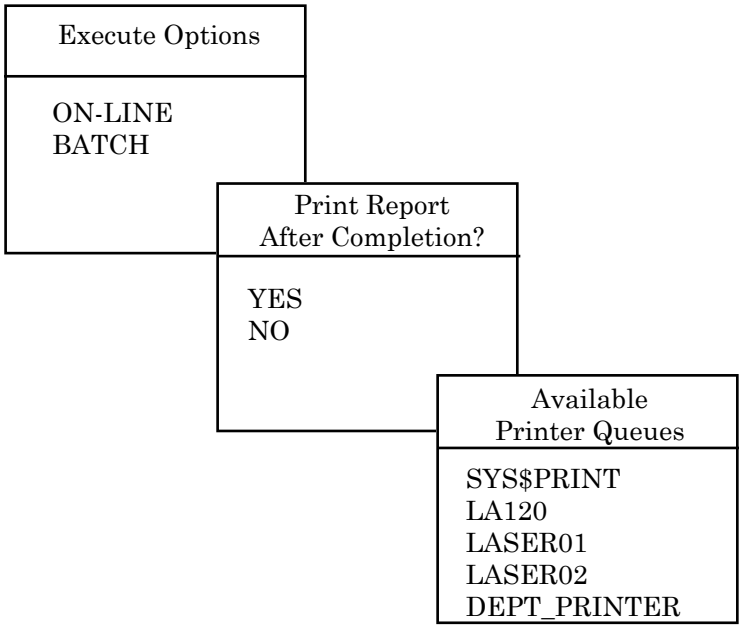
After specifying the format of the report, you may elect to run the report in **BATCH** or **ONLINE**. Running the report in **BATCH** frees up your screen to return to scanning the database or to specify a new report. For large databases and reports that include **DOMAINS** (including the **FULL** report), it is recommended that the report be run in batch.

PRINT_REPORT



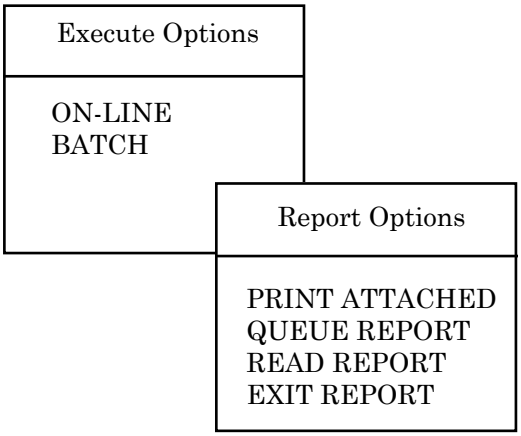
If you elect to run the report in **BATCH**, you may elect to print the report on completion. If you elect to print the report, you will be given a list of available printer queues to specify for the output.

AVAILABLE_QUEUES



This is a list of all available printer queues on your system. Be aware that the report(s) generated by DBAnalyzer require the printer be set to 132 column mode. Select your printer accordingly.

REPORT_OUTPUT_OPTIONS



PRINT ATTACHED will print the report to your attached printer port device.

QUEUE REPORT will allow you to print the report to one of the available printer queues on the system.

READ REPORT will put you into the **OpenVMS EDT** editor to peruse the output. To read the report in 132 column mode, after entering the file, press **<<PF1>>W**. An **EDTINI** file is included with DBAnalyzer that defines the following keys:

<<PF1>>W	Set screen 132
<<PF1>>N	Set screen 80
<<PF1>>R	Shift screen right
<<PF1>>L	Shift screen left
<<PF1>>Q	Quit out of editor

EXIT REPORT returns you to scan mode.

Chapter 2

DBXAct for Rdb

What is DBXAct?

DBXAct for Rdb is a monitoring performance tool designed to generate baseline statistics and reports on Rdb activity. DBXAct for Rdb allows you to clearly understand the hundreds of data points available for monitoring database activity. The detailed statistics and reports generated by DBXAct will allow you to perform precise tuning and optimization of your Rdb database.

Rdb Monitoring and Performance Tuning with DBXAct

Understanding Rdb activity is an important step in tuning your Rdb database. What type of activity, how much activity, and where the activity is occurring is vital information you need when optimizing the Rdb database and the system on which it runs.

Establishing benchmarks to accurately reflect the activity of your database before and after changes are made is critical in order to understand the effects of the changes made. Tuning your Rdb database without measurements is like working in the dark. You must be able to see the results of your changes in order to determine if your tuning efforts have been successful.

Tuning has been described as a balancing process. Resources available for a system are finite. Finding the optimal balance between available resources such as CPU, memory, and disk I/O is the key to successful Rdb database tuning and optimization. DBXAct for Rdb will accurately provide you with the information you need to successfully tune and optimize your Rdb database.

DBXAct Features

BaseLine Statistics A customizable report provides baseline measurements of activity.

Automatic Feed to DBTune by Tables, Indexes, and Clusters Information on total activity, I/O bias (read/write), and growth are available.

Monitoring Reporting of critical statistics for individual databases.

- DBXAct 6.0 supports Rdb versions through 7.2-x.
- Memory leaking has been eliminated, allowing DBXAct to execute for a long duration without resource conflicts.
- DBXAct collects extensive information on the Rdb database tables, indexes, clusters, Rdb version, average memory usage, buffer ratio, and session information. If the specific version of Rdb supports it, the statistics are collected for individual tables/indices, even if they reside within the same storage area and share the same physical files.
- Multiple report files are routed to one output file. This feature allows you to monitor several periods and project growth based on long-term monitoring. Average value column in the **REPORT** file is based on the average value per second instead of per scan.
- A DBXAct batch parameter selection is a recently added feature of DBXAct. A parameter is available to pass active hours per day for projection instead of 24 hours a day.
- You can shut down a specified DBXAct executing. This allows the remaining DBXAct executions to continue operating.

- Logical Symbols have now been corrected in **DBXACT.COM**.
- A registration ID code to compile and link is incorporated in the product.

DBXAct Overview and Concepts

DBXAct can monitor any active Rdb database on a system. An "active" database is one that either was opened manually via **RMU/OPEN** or that currently has users attached.

When executed, DBXAct generates a customizable summary report, highlighting critical statistics. Both summary and detailed information are available for the database.

A database activity file is also produced. This activity file can later be used by DBTune (a tuning tool) to evaluate the tuning needs of the database and create SQL scripts for its physical redesign. The activity file recommends a growth percentage based on the changes in record counts of the tables in the database, factored over a number of days that you specify. In addition to the growth percentage, tables are ranked according to the number and type of I/Os performed against them. A **READ** or **WRITE** bias affects the tuning calculations performed by DBTune. An example of one of these DBTune activity files can be found on the following page. The name of the DBTune file is always:

<database name>.DBTUNE

For example, if you have an Rdb database called MYDB, the activity file produced from a DBXAct run would be called:

MYDB.DBTUNE

Example of Database Activity

```
!
!       Filename:  mydb.dbtune
!
!
!       DBTune Activity Analysis File for Database:
!               dka100:[test72]mydb.rdb
!
!       File generated at 04/03/2006 15:39:35
```

```
!
!        Tuning Interval is 365 days.
!
|BEGIN SOURCE INFORMATION|
*SOURCE=DBXACT
*BEGAN= 04/03/2006 14:39:35
*ENDED= 04/03/2006 15:39:35
*Duration=       3600
*PROJECTION PERIOD=       365
*SYSTEM= HP ITA
|END SOURCE INFORMATION|
!
!        Database Information
!
|BEGIN DATABASE INFORMATION|
*DB_Name= mydb
*RDBMS_ENGINE= Rdb
*VERSION= 72(Multiversion)
*STORAGE_AREA=   10
*DB_SIZE=        2835647
*GB_SIZE=        250
*LB_SIZE=        20
*BUFFER_BLOCK_CNT=        6
*TABLES=         49
*INDICES=        57
*AVG_ROW_BYTES=  743
|END DATABASE INFORMATION|
!
!        Summary Statistics
!
|BEGIN SUMMARY STATISTICS|
*Total Read I/O=       630
*Total Write I/O=       210
|END SUMMARY STATISTICS|
!
!        Buffer Ratio Information
!
|BEGIN RATIO OBSERVED|
*Global Buffer Enabled=   N
*Buffer Hit Ratio=         93.97
|END RATIO OBSERVED|
!
!        Transaction Profile
!
|BEGIN TRANSACTION PROFILE|
*Tables Read=    349
*Tables Written= 0
*Records Read=   694
*Records Written=         0
*Clustered Records=       28
*Concurrent Users=        2
*Active Users=   1
|END TRANSACTION PROFILE|
!
!        Memory Usage Information
!
|BEGIN MEMORY USAGE INFORMATION|
*Average Memory Used=     1644
*Total Memory= 131024
*Available Memory= 35606
|END MEMORY USAGE INFORMATION|
!
!        Statistics Information
```

```
!
|BEGIN STATISTICS INFORMATION|
*Transactions=       420
*Verb Successes=     29820
*Verb Failures=         0
*Ckpts Due To Txn Lim=         0
*Ckpts Due To AIJ Growth=         0
*Ckpts Due To Time Lim=         0
*Data Read Time=         0
*Data Write Time=         1
*Locks Requested=     13670
*Locks Released=     13612
*Records Fetched=    170310
*Records Stored=         0
|END STATISTICS INFORMATION|
!
! Listed below are values that can be incorporated in tuning:
! The user may change values for those columns underlined with
! "--*--".  Listed below are reserved characters used by
! DBTune when interpreting this file:
! "!"  :  Comment line (if ! is the 1st char, line is ignored).
! "/"  :  Column separator, MUST exist between column values.
! "[]" :  The enclosed item is a database TABLE.
! "{}" :  The enclosed item is a database INDEX.
! "<>" :  The enclosed item is a database CLUSTER.
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!                                    Activity  %Growth  BIAS
!  [TABLE] or {INDEX} or <CLUSTER>     (1-9)     (0-999) (1..100)
! ---------------------------------  ---*----  ---*---  --*--
|BEGIN ITEM INFORMATION|
[ CANDIDATES                       ]  /   1    /    0   /   50
[ COLLEGES                         ]  /   1    /    0   /   50
[ DEGREES                          ]  /   1    /    0   /   50
[ DEPARTMENTS                      ]  /   1    /    0   /   79
[ EMPLOYEES                        ]  /   8    /    0   /  100
[ JOBS                             ]  /   1    /    0   /   50
[ JOB_HISTORY                      ]  /   1    /    0   /   50
[ RESUMES                          ]  /   1    /    0   /   50
[ SALARY_HISTORY                   ]  /   1    /    0   /   50
[ WORK_STATUS                      ]  /   1    /    0   /   50
{ COLL_COLLEGE_CODE                }  /   1    /
{ DEG_COLLEGE_CODE                 }  /   1    /
{ DEG_EMP_ID                       }  /   1    /
{ DEPARTMENTS_INDEX                }  /   9    /
{ EMPLOYEES_HASH                   }  /   1    /
{ EMP_EMPLOYEE_ID                  }  /   1    /
{ EMP_LAST_NAME                    }  /   1    /
{ JH_EMPLOYEE_ID                   }  /   1    /
{ JOB_HISTORY_HASH                 }  /   1    /
{ SH_EMPLOYEE_ID                   }  /   1    /
|END ITEM INFORMATION|
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

Getting Started

This section provides the information you need to quickly install and run DBXAct for Rdb. If you are familiar with installing and running DBXAct for Rdb, you may prefer to go directly to the **Quick Start** section below. Less experienced users should carefully read the remaining sections in this chapter, which provide detailed system requirements and installation instructions.

Quick Start

☐ **To install and run DBXAct for Rdb:**

1. Install DBXAct for Rdb. (If necessary, see the following sections on system requirements and installation.)

2. Go to the section on page 91, titled **Running DBXAct**.

System Requirements

Item	Requirement
Operating System	Open VMS 6.1 or higher. For sites where the version of VMS is lower than 6.1, possible solutions may be provided. Please contact ALI's technical support for assistance. Phone: (866) 257-8970
Disk Space	60,000 blocks of disk space .
Authorize Settings	See the following **AUTHORIZE** description.

Recommended **minimum AUTHORIZE** settings for a DBXAct user account. (Medium and large databases may need to increase these numbers.)

Username:	USER			Owner:	USER
Account:	USER			UIC:	[Group, Member]
CLI:	DCL			Tables:	DCLTABLES
Default:	<disk>:[dir]				
LGIMD:	LOGIN				
Login Flags:					
Primary Days:	Mon. Tues. Wed. Thurs. Fri.				
Secondary Days:	Sat. Sun.				
No access restrictions					
Expiration:	(none)	Pwdminimum:	0	Login Fails:	0
Pwdlifetime:	(none)	Pwdchange:			
Last Login:					
Maxjobs:	0	Fillm:	512	Bytlm:	10000
Maxacctjobs:	0	Shrfillm:	0	Pbytlm:	0
Maxdetach:	0	BIOlm:	100	Jtquota:	1024
Prclm:	4	DIOlm:	100	Wsdef:	1024
Prio:	4	ASTlm:	113	Wsquo:	5120
Queprio:	0	TQElm:	10	Wsextent:	5120
CPU:	(none)	Enqlm:	8000	Pqlfquo:	80000
Authorized Privileges:	GROUP TMPMBX NETMBX				
Default Privileges:	GROUP MPMBX NETMBX				

Warning If these minimums are not in place when DBXAct is executed, the analysis may fail!

Installing DBXAct for Rdb

> **Caution** If the minimum requirements listed in the System Requirements section are not available, DBXAct may fail when executed.

☐ **To install DBXAct for Rdb:**

1. Back up your system disk (optional).
2. Log in under the **SYSTEM** account or an account that has the VMS privilege **SYSPRV**.
3. Place the DBXAct distribution tape in the tape drive.
4. Type the following command to invoke the VMS install facility to install DBXAct on your system:

   For VAX/VMS

   ```
   $ @SYS$UPDATE:VMSINSTAL DBXRDBVMSØ60 <tape-drive>:
   ```

   For Alpha AXP:

   ```
   $ @SYS$UPDATE:VMSINSTAL DBXRDBAXPØ60 <tape-drive>:
   ```

   For Itanium:

   ```
   $ @SYS$UPDATE:VMSINSTAL DBXRDBITA060 <tape-drive>:
   ```

   where `<tape-drive>` is the name of the device where the DBXAct distribution tape has been mounted (such as `MUA6:`).

> **Caution** DBXAct V6.0 should not be installed in the same directory with any other product from ALI (such as DBTune).

5. After the VMS installation has completed, place the following line into the system startup command file

(**SYS$MANAGER:SYSTARTUP_VMS.COM**)
so that the required logical is set up when the system is rebooted:
$ DEFINE/SYSTEM/EXEC ALI_DBX_HOME <disk>:[dir]

where `<disk>` and `[dir]` are the names of the disk and directory to which DBXAct was installed, such as `$1$DUA1:[DBXRDBVMS50]` or `$1$DUA1:[DBXRDBAXP50]`.

6. Edit the **SYS$MANAGER:SYLOGIN.COM** file and add the following symbol:

```
$ DBXACT :== @ALI_DBX_HOME:DBXACT.COM
```

7. To obtain a license pak for DBXAct, type the following commands:

```
$ SET DEFAULT ALI_DBX_HOME

$ EDIT DBXACT.LICENSE
```

8. For each node ("machine") on which you wish to run DBXAct, do the following:

   - Replace "your node name" with the node name of the machine on which you have installed DBXAct. To get this information, type:

     ```
     $ WRITE SYS$OUTPUT F$GETSYI ("nodename")
     ```

   - If the "operating system" value supplied with your license is not accurate for your system, replace it with the output generated from the following command:

     ```
     $ WRITE SYS$OUTPUT F$GETSYI ("node_swtype")
     ```

   - Replace "your company name" with your company's full name.

   - Exit and save the file.

9. To obtain the appropriate REGISTRATION ID for each machine entered, call ALI at (866) 257-8970 [or (803) 648-5931] or fax a copy of the altered **DBXACT.LICENSE** file to (803) 641-0345. Be sure to indicate the version of the product you are using.

> **Note** International clients may obtain registration IDs or support through their local distributor's office.

Running DBXAct

This section describes how to start and run DBXAct, and explains the significance of all DBXAct variables.

Starting DBXAct

To invoke DBXAct, type the following:

```
$@ALI_DBX_HOME:DBXACT
```

A DCL symbol can be created to make this easier:

```
DBXACT :== "@ALI_DBX_HOME:DBXACT.COM"
```

This allows DBXAct to be executed by the following command:

```
$DBXACT
```

Check with your system manager to see if a symbol was created or create one in your own **LOGIN.COM**.

If you have several different versions of Rdb currently running on your system, you will now see the following paragraph:

```
******************************************************

       WELCOME TO DBXACT VERSION 6.0 FOR RDB!

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

The logical ALI_DBX_DATABASE is assigned to the database:

            <Value assigned to this logical.>

This will be the database monitored.
If the version of the database is not: <version #>, then enter
the proper version now. Otherwise hit <RETURN>.

Rdb Version:
```

If you only have one version running, you will see this:

```
******************************************************

       WELCOME TO DBXACT VERSION 6.0 FOR RDB!

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

The logical ALI_DBX_DATABASE is assigned to the database:

                <value assigned to this logical>

What version of Rdb is the database:
                                  <database name>?
```

After receiving the answers to these questions, the command file displays a paragraph describing the default parameter settings for the procedure:

```
                  Current DBXAct
                Parameter Settings

DBXAct will record statistics for 8 hours.
While executing, it will scan for statistics every 10 seconds.
DBTune activity files will project growth for 30 days.
DBXAct will execute in batch on queue SYS$BATCH.

Do you wish to make changes to any of these settings (Y/N)[Y]:
```

Entering a **CTRL-Z** at this prompt lets you exit the procedure without running DBXAct. By entering a **Y** the procedure will bring up a menu of parameter options that you can change:

```
                    DBXAct Batch
                Parameter Selection

1.  Monitoring Duration
2.  Monitoring Scan Rate
3.  Growth Projection Interval
4.  Batch Queue

Enter the number you wish to change [0]:
```

Entering **CTRL-Z** or a zero at this prompt returns you to the parameter paragraph. By selecting a number from one to four, the procedure will prompt you for changes to be made to parameters that govern DBXAct's execution. Following is a sample of the prompts that you will see for each option. Please note that entering a simple **Return** will always result in the use of the current value.

```
1.  Monitoring Duration
    DBXAct will monitor database activity for 8 hours.
    How many hours would you like to monitor activity (0,720) [8]:

2.  Monitoring Scan Rate
    Current scan rate is 10 seconds.
    The valid range is 5 to 3600.
    Enter new scan rate (5,3600) [10]:

3.  Growth Projection Interval
    Current growth projection interval is for 30 days.
    The valid range is 1 to 365.
    Enter new growth interval (1,365) [30]:

4.  Batch Queue
    DBXAct will be submitted to execute in queue SYS$BATCH.
    Enter the name of the queue, or 'HELP' for a list of queues.
```

Any batch run of DBXAct will produce a DBTune dynamic activity file called **<database_name>.DBTUNE** and a report file called **<database name>.REPORT** in **SYS$LOGIN**. Two log files, **DBX_BATCH.LOG** and **DBXACT.LOG**, will also be created in **SYS$LOGIN**. If you suspect something may have gone wrong during the DBXAct process, be sure to check the log file in this directory.

Explanation of DBXAct Variables

Monitoring Duration The default duration period for DBXAct monitoring is eight hours. This can be changed to any desired amount of time, given that the specified time is greater than or equal to one hour and that only full hour variables are given. (Two and a half hours is not allowed, but two hours or three hours is allowed.)

Monitoring Scan Rate This value specifies the interval at which to scan. The default value is to scan every ten seconds, but this may be changed to every thirty seconds or every five seconds. By specifying a value of x seconds, DBXAct will scan the database, then "sleep" for x seconds, then scan the database, then "sleep" for x seconds, and so on. A value of zero is not allowed.

Growth Projection Interval The default value for the growth projection interval is set to thirty days. This means that the DBTune file will project the results of the monitoring session over thirty days, rather than simply the monitoring duration period. For example, if DBXAct is set to monitor a database for 24 hours and a table grows by 1 percent during that time period, then the 1 percent growth would be projected over a 30-day time period to result in 30 percent growth for that table. Use this parameter to indicate how many days will elapse between database tuning sessions so that appropriate growth can be planned for and used to anticipate database storage needs.

Batch Queue This is the name of the batch queue where DBXAct will run. The default value is **SYS$BATCH**, but can be changed to any logical name for a batch queue.

Using DBXAct

Note DBXAct can only be run in batch mode.

The DBXAct process collects database statistics over a specified period of time, and later these statistics are reported in DBXAct's customizable summary report.

The performance of DBXAct can be controlled with a number of runtime logicals or variables. The section about DBXAct Logicals beginning on page 103 lists the logical names that affect DBXAct.

Important The account used to run DBXAct must have sufficient privileges to perform an **RMU/SHOW STATISTICS** for the database and to select data from **SYSTEM** relations. The VMS privilege **SYSPRV** will be sufficient or a combination of database privileges.

The current version(s) of DBXAct support Rdb 4.2-13 through Rdb 7.2-x.

Analyzing Data Gathered with DBXAct

This section provides information on the various database statistics gathered by DBXAct for Rdb. These statistics will provide you with important information that will allow you to accurately tune and optimize your Rdb database.

Storage Area Statistics

Storage area statistics are always reported in the DBXAct customizable report. These are used to determine which storage areas are more active than others, and where any trouble areas may arise. The following statistics for each storage area are reported on a per I/O basis and can be used for comparison purposes:

Read IOs
Write IOs
Extend IOs
Read Blocks
Write Blocks
Extend Blocks
Read Stall Time
Write Stall Time
Extend Stall Time

General Database Statistics

Transactions
Verb Successes
Verb Failures
in GB, Rt Ver, Need Lck

Stall Statistics

Data Read Time
Data Write Time

Locking Summary Statistics

Locks Requested
Locks Unlocked
Locks Promoted
Locks Demoted
Requests Stalled
Request Deadlocks
Proms Stalled
Prom Deadlocks
Stall Time

PIO Data Fetch Statistics

PIO Fetches for IO: Sum of PIO fetches for reads and PIO fetches for writes

PIO Spam Fetches for IO: Sum of PIO Spam fetches for reads and PIO Spam fetches for writes

Record Statistics

Records Fetched
Records Fragmented
Records Stored
Pages Checked

Index Statistics

B-tree Scans Number of B-tree scans during monitoring period

Hash Scans Number of hash scans during monitoring period

Checkpoint Statistics

Ckpts Due to Txn Lim
Ckpts Due to AIJ Growth
Ckpts Due to Time Lim

Transaction Statistics

Tot Txn Durtn for Txns
Tot Txn Durtns

Summary All File IOs

All Files Read IOs
All Files Write IOs
All Files Extend IOs
All Files Read Blks
All Files Write Blks
All Files Extend IOs
All Files Read Stall Time
All Files Write Stall Time
All Files Extend Stall Time

Generating and Understanding Reports

DBXAct for Rdb produces a report titled **<database name>.REPORT**. This outlines summary and detailed information for the various statistics it monitors. The report is produced automatically at the end of each monitoring run, after data has been gathered. You can customize the report to generate information about statistics of interest using the following file:

ALI_DBX_HOME:STAT_PROFILE.EDT

File Statistics

By default, ten important statistics are selected in the **<database name>.REPORT** file. In addition to these ten statistics, information detailing file I/O, file blocks transferred, and file stall time is always reported. Information about reads, writes, and extends is recorded about each of these three categories. An example report file will look like the following:

```
File Name:       mydb.report

                        DBXAct - Summary Report for dka100:[test72]mydb.rdb
                        ---------------------------------------------------------

        Start Date/Time: 04/03/2006 14:39:35       End Date/Time: 04/03/2006 15:39:35

        Total System Memory: 131024                Available System Memory: 35606
        Buffer Hit Rate (%):  93.97                Global Buffers Enabled: N

                                  Database Statistics
                                  -------------------

  Statistic Name                 Total       Avg Value     Peak Value  Day/Time
---------------------------    -----------   ----------    -----------  --------
Transactions                           420         0.12            321  04/03 14:54
Data Read Time                           0         0.00              0  04/03 15:39
Data Write Time                          1         0.00              1  04/03 14:54
Locks Requested                      13670         3.80          10417  04/03 14:54
Locks Released                       13612         3.78          10256  04/03 14:54
Stall Time                               0         0.00              0  04/03 15:39
Records Fetched                     170310        47.31         130556  04/03 14:54
Records Fragmented                   10080         2.80           7728  04/03 14:54
AIJ File Asynchronous Read               0         0.00              0  04/03 15:39
DBS File Asynchronous Read           10710         2.97           8211  04/03 14:54
ROO File Asynchronous Read               0         0.00              0  04/03 15:39
RUJ File Asynchronous Read               0         0.00              0  04/03 15:39
ACE File Asynchronous Read               0         0.00              0  04/03 15:39
User Sessions                           20         0.01              2  04/03 14:44
All Files Read IOs                     630         0.17              0  04/03 15:39
All Files Write IOs                    210         0.06              0  04/03 15:39
```

File I/O Statistics

File Name	Read I/O	Write I/O	Extend I/O	Total I/O	% of Total I/O
RDB$SYSTEM	420	0	0	420	50.00
EMPIDS_LOW	0	0	0	0	0.00
EMPIDS_MID	0	0	0	0	0.00
EMPIDS_OVER	0	0	0	0	0.00
DEPARTMENTS	210	210	0	420	50.00
SALARY_HISTORY	0	0	0	0	0.00
JOBS	0	0	0	0	0.00
EMP_INFO	0	0	0	0	0.00
RESUME_LISTS	0	0	0	0	0.00
RESUMES	0	0	0	0	0.00
Total	630	210	0	840	
Pct of Total	75.00%	25.00%	0.00		

File Blocks Transferred

File Name	Read Blocks	Write Blocks	Extend Blocks	Total Blocks	% of Total Blocks
RDB$SYSTEM	1680	0	0	1680	40.00
EMPIDS_LOW	0	0	0	0	0.00
EMPIDS_MID	0	0	0	0	0.00
EMPIDS_OVER	0	0	0	0	0.00
DEPARTMENTS	1260	1260	0	2520	60.00
SALARY_HISTORY	0	0	0	0	0.00
JOBS	0	0	0	0	0.00
EMP_INFO	0	0	0	0	0.00
RESUME_LISTS	0	0	0	0	0.00
RESUMES	0	0	0	0	0.00
Total	2940	1260	0	4200	
Pct of Total	70.00%	30.00%	0.00%		

File Stall Time

File Name	Read Stall Tm	Write Stall Tm	Extend Stall Tm	Total Stall Tm	% of Tot Stl Tm
RDB$SYSTEM	0	0	0	0	0.00
EMPIDS_LOW	0	0	0	0	0.00
EMPIDS_MID	0	0	0	0	0.00
EMPIDS_OVER	0	0	0	0	0.00
DEPARTMENTS	0	78	0	78	100.00
SALARY_HISTORY	0	0	0	0	0.00
JOBS	0	0	0	0	0.00
EMP_INFO	0	0	0	0	0.00
RESUME_LISTS	0	0	0	0	0.00
RESUMES	0	0	0	0	0.00
Total	0	78	0	78	
Pct of Total	0.00%	100.00%	0.00%		

Table Statistics

Table Name	Read Activity	Write Activity	Total Activity	% of Total
CANDIDATES	0	0	0	0.00
COLLEGES	0	0	0	0.00
DEGREES	0	0	0	0.00
DEPARTMENTS	24000	6240	30240	14.38
EMPLOYEES	180000	0	180000	85.62
JOBS	0	0	0	0.00
JOB_HISTORY	0	0	0	0.00
RESUMES	0	0	0	0.00
SALARY_HISTORY	0	0	0	0.00
WORK_STATUS	0	0	0	0.00
Total	204000	6240	210240	
Pct of Total	97.03%	2.97%		

Index Statistics

Index Name	Read Activity	Write Activity	Total Activity	% of Total
COLL_COLLEGE_CODE	0	0	0	0.00
DEG_COLLEGE_CODE	0	0	0	0.00
DEG_EMP_ID	0	0	0	0.00
DEPARTMENTS_INDEX	480	0	480	100.00
EMPLOYEES_HASH	0	0	0	0.00
EMP_EMPLOYEE_ID	0	0	0	0.00
EMP_LAST_NAME	0	0	0	0.00
JH_EMPLOYEE_ID	0	0	0	0.00
JOB_HISTORY_HASH	0	0	0	0.00
SH_EMPLOYEE_ID	0	0	0	0.00
Total	480	0	480	
Pct of Total	100.00%	0.00%		

Customized Reports

DBXAct monitors many types of statistics. These statistics provide valuable information regarding database activity and can be used as a tool in improving performance. DBXAct's summary report furnishes detailed and summary information on any statistic selected in the file:

ALI_DBX_HOME:STAT_PROFILE.EDT.

Ten critical statistics are selected by default, but DBXAct provides a simple method for altering these statistics. Simply editing the file **STAT_PROFILE.EDT** and placing an '*****' directly to the left of any statistic causes DBXAct to report this statistic. The file **STAT_PROFILE.EDT** that was used to produce the above report is on the next page in the section titled "**Report Data**." Notice that there are no spaces between the '*****' and the statistic name, and that the '*****' is the only character that needs to be added. Removing the '*****' causes the statistic to *not* be included in the reports.

For each selected statistic, DBXAct reports its total change value over the monitoring period, its average value, peak value, and the time at which that peak occurred.

Report Data

Since DBXAct records cumulative values, taking the difference between data points during a time increment and adding the differences creates the report.

Example of the file **STAT_PROFILE.EDT:**

> **Note** Only those statistics proceeded by an "*****" character will be collected during the monitoring session.

```
*Transactions
Verb Successes
Verb Failures
Ckpts Due to Txn Lim
Ckpts Due to AIJ Growth
Ckpts Due to Time Lim
In GB, Rt Ver, Need Lck
Tot Txn Durtn for Txns
Tot Txn Durtns
*Data Read Time
*Data Write Time
*Locks Requested
Locks Unlocked
Locks Promoted
Locks Demoted
Requests Stalled
Request Deadlocks
Proms Stalled
Prom Deadlocks
*Stall Time
PIO Fetches For IO
PIO Spam Fetches For IO
*Records Fetched
```

```
*Records Fragmented
Records Stored
Pages Checked
B-tree Scans
Hash Scans
*All Files Read IOs
*All Files Write IOs
All Files Read Blks
All Files Write Blks
All Files Read Stall Time
All Files Write Stall Time
```

DBXAct Logical Names

ALI_DBX_HOME

This logical points to the "home" area for DBXAct. By home, we mean the disk and directory where the DBXAct executable and other distribution files reside. By modifying the system startup command file as specified in the installation instructions, this logical will be reassigned each time the system is rebooted.

ALI_DBX_DATABASE

This logical points to the database that will be monitored by DBXAct. It should be set at the **PROCESS** level. The value assigned to this logical should be of the form:

```
<DISK>:[<DIRECTORY>]<DATABASE>.RDB.
```

DBXACT$SHUTDOWN_FLAGS

This logical can be set at the **SYSTEM** level to force a particular DBXAct job to abort before its specified end time. If for some reason you or the system manager needs to stop DBXAct while it is in the middle of a long monitoring operation, assigning the value of the **PID** for the DBXAct process to this logical will cause the DBXAct process to complete processing whenever it wakes up from its hibernation or wait state, after the next scan. Therefore, although the DBXAct process may not stop immediately after setting this logical, it will begin to shut down after completing the next database scan.

Use of this logical is preferable to stopping the process and allows DBXAct to complete the creation of the DBTune activity file.

Answers to Commonly Asked Questions

Why does DBXAct run only in batch mode?

DBXAct simply collects statistics and reports them after the monitoring period in both the report file and the DBTune output file. Therefore, it only needs to be executed in batch mode.

Can DBXAct monitor multiple databases at the same time?

Not in the current release, but future releases will enable this characteristic. Several different DBXAct processes can be started, however. In this way, each process can monitor a different database.

What happens if the process is shut down while DBXAct is running?

If you need to stop DBXAct while it is in the middle of a long monitoring operation, you should follow these steps:

1. Use the **DCL** command

   ```
   show sys/page
   ```

   to locate the **PID** of the DBXAct process to be stopped. Note the **PID** number.

2. Use the **DCL** command

   ```
   define/system DBXAct$shutdown_flags #
   ```

   where # is the **PID** of the DBXAct process you wish to stop.

   Example assignment of the DBXACT$SHUTDOWN_FLAGS:

   ```
   $ define/system DBXAct$shutdown_flags 25A02703
   ```

How can I use Activity values to tune my database?

Philosophies vary in this area; some suggest tuning based on 50 to 150 percent of average activity, while others suggest 60 to 100 percent of peak, and still others suggest tuning to available resources. No single answer is more correct than the others are, and either of the first two choices must be weighed against the resources of the system.

What Rdb versions will DBXAct monitor?

DBXAct supports versions of Rdb from 4.2-13 through Rdb 7.2.

Chapter 3

DBTune for Rdb

What is DBTune for Rdb?

DBTune for Rdb is intended to maximize your Rdb performance without requiring excessive effort. To gather information necessary for the tuning process, it considers both:

1. The existing Rdb design (both physical and logical), and

2. Performance Analysis Data.

Thus, both the database design and its actual usage are incorporated in the database optimization procedure, typically resulting in a 30 to 50 percent improvement in performance (direct I/O, CPU usage, and elapsed time).

Using the logical and physical data gathered from the database, DBTune creates a Performance Analysis Data (PAD) file. This PAD file contains volume, workload, and environment information. You may customize this information with additional transaction activity. Additionally, the PAD file can be supplemented with a dynamic analysis of observed transaction data, which you can create manually or automatically using a third-party tool. The tuning process then combines the Performance Analysis data, the logical and physical Rdb design, and the transaction analysis results to create an optimized design.

DBTune achieves this transformation of the Rdb database without affecting the programs and queries that access it. Rather, the Rdb database is converted to a more efficient physical structure. The transformation takes advantage of the high-performance capabilities of Rdb, including the use of multi-file structures. Each storage area is tuned so that the appropriate page size and allocation are assigned according to the analysis inputs provided. Sorted, sorted ranked, and hashed indexes are stored so as to maximize throughput and minimize lock contention. Additionally, database parameters are tuned and storage areas are distributed across available disk drives to take advantage of system capacity for the particular physical design.

To assist with Rdb tuning plans, DBTune provides an objective measure of a database's complexity and tune rating. Empirical data indicates that databases with a complexity rating of at least 15 to 25 can achieve significant performance improvement through increase in complexity and the increase in database usage (e.g., more users, reports, etc.).

New & Enhanced Features to DBTune V6

Note DBTune 6.0 works with Rdb 6.1 or higher. You must use DBTune 4.0 if you are running an earlier version of Rdb.

Native C

DBTune V6 has been completely rewritten in native "C" code. This reduces memory. disk space, and system resource usage by eliminating previously used "third party" components.

Enhanced Disk Space Calculations

DBTune V6 takes into account the free space available on each specified disk or "device". Additionally, the space that is currently occupied, but which will be "freed", as part of the tuning process (ie space taken by a table that will be 'dropped' as part of the tuning process) is also taken into account.

The number of "devices" which can be specified for file placement can now been increased to 999.

DBTune will also now take into account the disk space allocated across DBDISKS even if multiple DBDISKs resolve to the same physical device.

Two overflow areas now exist: The disk:[dir] where the database "root" file is located, and the second is the default directory from which DBTune is being run.

Enhanced Sizing Calculations

DBTune V6 page size, threshold, and allocation calculations have been reviewed and updated to more accurately fit the data in question.

Default Values

DBTune V6 preserves column default values defined at the domain, and at the column level. Previous versions retained only the default values defined at the domain level.

Improved Tuning Strategy Behavior

"Existing" and "Relocate" strategies have been updated.

"Existing" strategy now keeps all existing areas and map names as they were found (with the exception of any new list areas or new storage areas for non-partitioned objects, that were previously located in the RDB$SYSTEM area). Areas are recreated in their original locations by default, although allocations, page sizes, and threshold values may be recalculated.Objects previously grouped together in an area will remain grouped (unless manually specified otherwise by the operator using the MODPAD file). Empty storage areas are NOT automatically dropped with "Existing" or "Relocate", as in previous versions of DBTune (as they may be intentionally empty for "on the fly" table creation). Partitioned objects remain in their existing locations, even if found in RDB$SYSTEM.

"Relocate" encompasses all the features of "Existing", but allows ALL the current storage area files to be redistributed across different disks. Previously, only the root file was relocated.

Improved Storage Area and Map Naming Support

DBTune V6 preserves the existing names of storage areas and storage maps whenever possible. When creating "New" areas/maps, the named objects are based on a clearer naming convention using the name of the associated table or index in question.

SQLNet for Rdb

DBTune V6 supports the presence of SQLNet for Rdb when tuning with RMU.Previously, a known problem (in some versions of Rdb) with Rdb's sequencing of object interdependencies required manual intervention to the DBTune scripts. SQLNet for Rdb objects are now ignored during tuning. Import/Export commands (part of Rdb's SQL syntax) may still have this sequencing problem. It depends on the specific version of Rdb, and is reportedly in the process of being corrected by Rdb Engineering. If you are unsure of the status of your particular version of Rdb, you should check with Rdb Engineering or contact ALI for more information. SQLNet for Rdb objects are typically stored in RDB$SYSTEM, and are not moved by DBTune.

Temporary Tables

DBTune 6.0 correctly preserves both global and local temporary tables. Temporary tables, a new feature in Rdb 7, provide the convenience of storing and manipulating short-term data in a table, rather than using a flat file or repeatedly creating and then dropping a table. Temporary tables can be used to store the output of a query or other intermediate results. Global temporary tables allow data to be shared between different modules in a single SQL session. Local temporary tables do not allow data to be shared. Metadata for both global and local temporary tables is stored in the database and persists beyond a SQL session. Data in temporary tables does not persist beyond the SQL session.

Since the data in the temporary table does not exist after the SQL session, temporary tables will always be empty after tuning. DBTune 6.0 correctly preserves any global or local temporary tables present in an Rdb 7 database.

Partitioning

Vertical and horizontal partitioning is used to distribute portions of a table/indice across multiple storage areas. The two partitioning methods may be used individualy, or together, for both tables and indices. This can be driven by any one of several needs such as to 'distribute' a large table/index across multiple disks due to storage constraints, to distribute "hot spots", or to distribute data in a more "organized" manner (ie by company, department, etc).

DBTune V6 supports the preservation of existing vertical and horizontal partitioning. Existing partitions are resized, and storage parameters are tuned based on the specific type of data and number of records, associated with each individual partition being tuned.

Partitioning is an extremely application specific feature, and as such DBTune relies on the DBA and/or application designer to have previously assigned the proper limits to the partitioned objects.

Row Cache

Row cache allows the most frequently accessed rows of a logical or physical database area to remain in memory even after the associated pages are flushed back to the disk. Row cache can be used on single node VAX and AXP systems and can significantly improve performance if memory is ample and most database accesses involve a limited number of table or index rows.

DBTune V6 supports the preservation of existing read/write row caching.

Caching is an extremely application specific feature, and as such DBTune relies on the DBA and/or application designer to have previously assigned the proper limits to the cached objects.

SQL92 & SQL99 Support

DBTune V6 supports older Rdb databases which were originally defined using the SQL92 standard (common for Rdb V6 and the previous DBTune default) and which have subsequently been upgraded to Rdb 7, as well as 'NEW' databases created using the current Rdb SQL99 default syntax.

Databases which have been created using the earlier SQL92 standard, or have been successfully tuned with earlier versions of DBTune, should consider defining the VMS logical ALI_SQL_RULES to "SQL92" prior to running DBTune V6. This is most commonly required when reserved words were used as identifiers (ie column names for example), and were enclosed in " " when originally defined (ie using "year" as a column name).

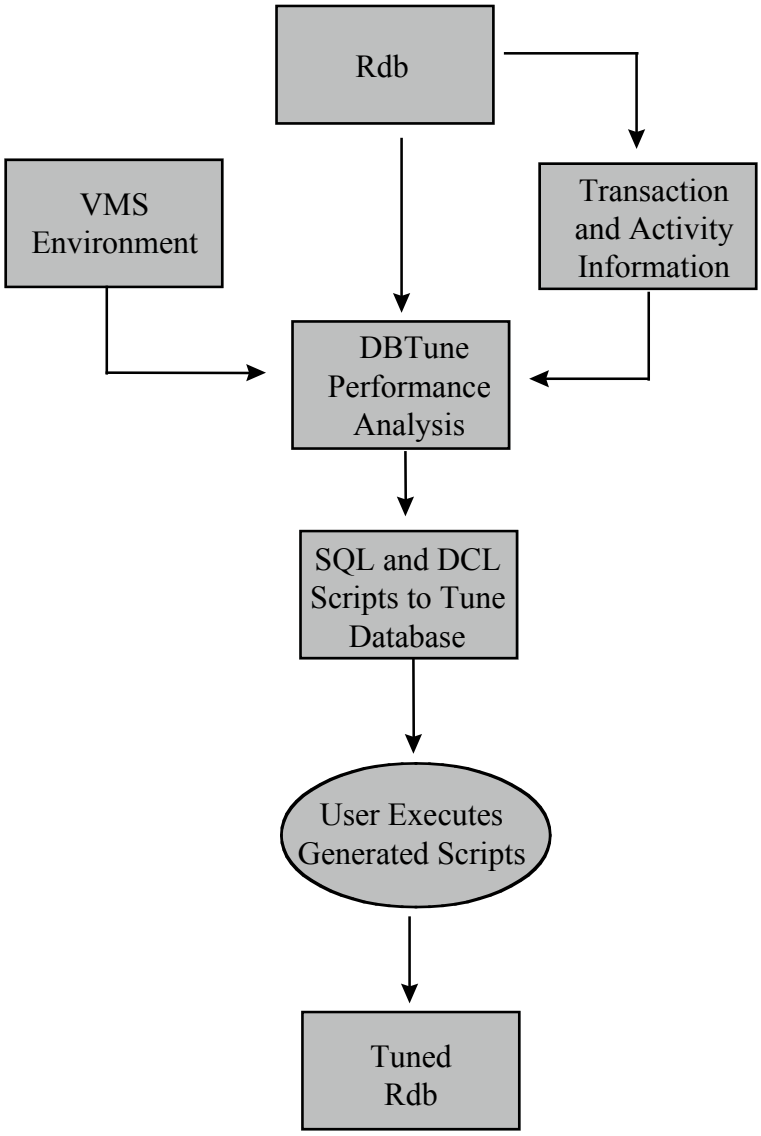
Sorted Ranked Indices

Rdb 7 supports a ranked B-tree structure for sorted indexes. Sorted ranked indexes allow better optimization of queries, especially queries involving range retrievals. The sorted ranked index type can also reduce lock contention and disk I/O. DBTune 6.0 preserves any sorted ranked indexes while running an Rdb 7.0 database, and allows sorted and hashed indices to be converted to/from sorted ranked with appropriate tuning parameters being calculated for all conversions.

New Logical Names

Earlier versions of DBTune utilized a logical name " convention" that dated bach to a very early version of the product known as FREND (aka Front End) , of which DBTune was only a small part. The "FRENDXXXXXX" logical name convention originated at that time, and was supported for many years. Now that ALI has taken over support for the DBTune product and tools, and the FREND product is no longer supported, a new naming convention has been implemented. As a general rule, logicals which were phrased as "FRENDXXXXXX" are now "ALI_XXX_XXX" (ie FRENDRDBIMPORT is now ALI_RDB_IMPORT).

DBTune Considers
Dynamic Workload, Volume, and Environment Inputs
During Its Performance Analysis



Getting Started

This section provides the information needed for you to quickly install and run DBTune. It also includes system requirements necessary to run the application.

DEC VAX/OpenVMS

Recommended **minimum AUTHORIZE** settings for a DBTune user account. (Medium and large databases may need to increase these numbers.)

Username:	USER			Owner:	USER
Account:	USER			UIC:	[Group, Member]
CLI:	DCL			Tables:	DCLTABLES
Default:	<disk>:[dir]				
LGIMD:	LOGIN				
Login Flags:					
Primary Days:	Mon. Tues. Wed. Thurs. Fri.				
Secondary Days:	Sat. Sun.				
No access restrictions					
Expiration:	(none)	Pwdminimum:	0	Login Fails:	0
Pwdlifetime:	(none)	Pwdchange:			
Last Login:					
Maxjobs:	0	Fillm:	512	Bytlm:	120000
Maxacctjobs:	0	Shrfillm:	0	Pbytlm:	0
Maxdetach:	0	BIOlm:	100	Jtquota:	1024
Prclm:	4	DIOlm:	100	Wsdef:	1024
Prio:	4	ASTlm:	113	Wsquo:	5120
Queprio:	0	TQElm:	10	Wsextent:	5120
CPU:	(none)	Enqlm:	8000	Pqlfquo:	160000
Authorized Privileges:	GROUP TMPMBX NETMBX				
Default Privileges:	GROUP MPMBX NETMBX				

Warning If these minimums are not in place when DBTune is executed, the tuning process may fail!

Note See the **REVIEW_AND_GUIDE.REPORT** file created by DBTune for suggested **AUTHORIZE** settings for individual database user accounts.

Below are **MINIMUM** settings recommended for several **SYSGEN** parameters. If changes are made to any of the **SYSGEN** parameters listed below, your system will need to be **REBOOTED** to make the changes effective. It is recommended that **AUTOGEN** be used to make any required changes.

SYSTEM PARAM	MINIMUM SETTING	DESIRED SETTING
CHANNELCNT *	512	2047
VIRTUALPAGECNT *	160000	240000
LOCKIDTBL **	2048	10240
LOCKIDTBL_MAX **	2048	61440
RESHASHTBL **	512	2560
PROCSECTCNT	64	64
GBLPAGES *	100000	200000
GBLSECTIONS	600	600
GBLPAGFIL ***	50000	100000
CTLPAGES ****	100	200

(*) The values for these parameters may need to be increased if using Rdb **Global Buffers**. Please see the **REVIEW_AND_GUIDE.REPORT** generated by DBTune for more details on a particular database.

(**) The setting for **LOCKIDTBL** must be four (4) times the setting for **RESHASHTBL**. If you change one setting, you should change the other as well. The setting for **LOCKIDTBL_MAX** must be equal to or greater than the setting for **LOCKIDTBL**.

(***) Total system **PAGE FILE** space must be larger than the setting for **GBLPAGFIL**. Thus, if **GBLPAGFIL** is increased, ensure that adequate **PAGEFILE.SYS** space exists. You can view the current **PAGE FILE** sizes on a system with the DCL command: **`$ SHOW MEMORY/FILES`**.

(****) The setting for **CTLPAGES** is recommended but not required. However, if logicals are being used to specify database storage areas, the suggested setting for **CTLPAGES** should be adhered to.

DEC AXP/OpenVMS& I64 OpenVMS

Recommended **minimum AUTHORIZE** settings for a DBTune user account. (Medium and large databases may need to increase these numbers.)

Username:	USER			Owner:	USER
Account:	USER			UIC:	[Group, Member]
CLI:	DCL			Tables:	DCLTABLES
Default:	<disk>:[dir]				
LGIMD:	LOGIN				
Login Flags:					
Primary Days:	Mon. Tues. Wed. Thurs. Fri.				
Secondary Days:	Sat. Sun.				
No access restrictions					
Expiration:	(none)	Pwdminimum:	0	Login Fails:	0
Pwdlifetime:	(none)	Pwdchange:			
Last Login:					
Maxjobs:	0	Fillm:	512	Bytlm:	120000
Maxacctjobs:	0	Shrfillm:	0	Pbytlm:	0
Maxdetach:	0	BIOlm:	150	Jtquota:	1024
Prclm:	4	DIOlm:	150	Wsdef:	2000
Prio:	4	ASTlm:	250	Wsquo:	4096
Queprio:	0	TQElm:	10	Wsextent:	16384
CPU:	(none)	Enqlm:	8000	Pqlfquo:	160000
Authorized Privileges:	GROUP TMPMBX NETMBX				
Default Privileges:	GROUP MPMBX NETMBX				

Warning If these minimums are not in place when DBTune is executed, the tuning process may fail!

Note See the **REVIEW_AND_GUIDE.REPORT** file created by DBTune for suggested **AUTHORIZE** settings for individual database user accounts.

Below are **MINIMUM** settings recommended for several **SYSGEN** parameters. If changes are made to any of the **SYSGEN** parameters listed below, your system will need to be **REBOOTED** to make the changes effective. It is recommended that **AUTOGEN** be used to make any required changes.

SYSTEM PARAM	MINIMUM SETTING	DESIRED SETTING
CHANNELCNT *	512	2047
VIRTUALPAGECNT *	160000	240000
LOCKIDTBL **	2048	10240
LOCKIDTBL_MAX **	2048	61440
RESHASHTBL **	512	2560
PROCSECTCNT	64	64
GBLPAGES *	100000	200000
GBLSECTIONS	600	600
GBLPAGFIL ***	50000	100000
CTLPAGES ****	100	200

(*) The values for these parameters may need to be increased if using Rdb Global Buffers. Please see the **REVIEW_AND_GUIDE.REPORT** generated by DBTune for more details on a particular database.

(**) The setting for **LOCKIDTBL** must be four (4) times the setting for **RESHASHTBL**. If you change one setting, you should change the other as well. The setting for **LOCKIDTBL_MAX** must be equal to or greater than the setting for **LOCKIDTBL**.

(***) Total system **PAGE FILE** space must be larger than the setting for **GBLPAGFIL**. Thus, if **GBLPAGFIL** is increased, ensure that adequate **PAGEFILE.SYS** space exists. You can view the current **PAGE FILE** sizes on a system with the DCL command: **`$ SHOW MEMORY/FILES`**.

(****) The setting for **CTLPAGES** is recommended but not required. However, if logicals are being used to specify database storage areas, the suggested setting for **CTLPAGES** should be adhered to.

Installing DBTune

To install DBTune for Rdb from a tape drive:

1. Back up your system disk (optional).
2. Log in under the SYSTEM account.
3. Put the DBTune distribution tape in the tape drive.
4. Type in the following DCL command to invoke the VMS install facility to install DBTune on your system:

   For VAX/VMS

   ```
   $ @SYS$UPDATE:VMSINSTAL DBTRDBVMS60 <<tape-drive>>:
   ```

   For Alpha AXP

   ```
   $ @SYS$UPDATE:VMSINSTAL DBTRDBAXP60 <<tape-drive>>:
   ```

   For Itanium I64

   ```
   $ @SYS$UPDATE:VMSINSTAL DBTRDBITA60 <<tape-drive>>:
   ```

   where <<tape-drive>> is the name of the device where the DBTune distribution tape has been mounted (e.g., MUA6:).

Note DBTune V6.0 should NOT be installed in the same directory with other versions of DBTune or any other product from ALI (i.e., DBAnalyzer).

5. After the VMS install has completed, place the following lines into the system startup command file (SYS$MANAGER:SYSTARTUP_VMS.COM) so that required logicals are set up when the system is rebooted:

```
$ DEFINE/SYSTEM/EXEC ALI_DBTUNE_HOME   <<disk>>:[dir]
$ DEFINE/SYSTEM/EXEC ALI_DBTUNE_SCRATCH
   <<disk>>:[dir.SCRATCH]
```

   where <<disk>> and [dir] are the disk and directory to which DBTune was installed (e.g., 1DUA1:[DBTRDBVMS60], 1DUA1:[DBTRDBITA60, or 1DUA1:[DBTRDBAXP60]).

6. Now, to obtain a license pak for DBTune, type in the following commands:

```
$ SET DEFAULT ALI_DBTUNE_HOME

$ EDIT DBTUNE.LICENSE
```

   For each node (“machine”) on which you wish to run DBTune:

   - Replace “your node name” with the node name of the machine on which you have installed DBTune. To get this information, type:

     ```
     $ WRITE SYS$OUTPUT F$GETSYI (“nodename”)
     ```

   - If the “operating system” value supplied with your license is not accurate for your system, replace it with the output generated from the following command:

     ```
     $ WRITE SYS$OUTPUT F$GETSYI (“node_swtype”)
     ```

   - Replace “your company name” with your company’s full name.

   - Exit and save the file.

To obtain the appropriate registration ID for each machine entered, call ALI at (866) 257-8970 [or (803) 648-5931], or fax a copy of the altered DBTUNE.LICENSE file to (803) 641-0345. International clients may also obtain registration IDs or support through their local distributor's office.

To install DBTune for Rdb from a CD-ROM:

1. Mount the CD using a command like

   ```
   $ MOUNT/OVER=ID <cd_device>:
   ```

2. Install the product with the command

   ```
   $ @SYS$UPDATE:VMSINSTAL <product_name>
   <cd_device>:[INSTALL]
   ```

   where <product_name> is the product you wish to install. For example:

   ```
   $ @sys$update:vmsinstal DBTRDBVMS060 dka400:[INSTALL]
   ```

Using DBTune

DBTune requires that you have either the VMS privilege **SYSPRV** or the appropriate database and **RMU** privileges. If you do not have **SYSPRV** privilege, DBTune requires the following **RMU** privileges:

"RMU$BACKUP," "RMU$UNLOAD," "RMU$LOAD," "RMU$DUMP," "RMU$OPEN," and "RMU$ANALYZE"

Note.DBTune 6.0 requires that the database to be tuned be Rdb 6.1 or higher

Online Use of DBTune

To run DBTune online:

```
$ @ALI_DBTUNE_HOME:DBTUNE.COM
```

Note Setting up a VMS symbol can make this easier.

```
$ DBTUNE :== "@ALI_DBTUNE_HOME:DBTUNE.COM"
```

allowing you to execute DBTune by typing:

```
$ DBTUNE
```

Batch Use of DBTune

To run DBTune in BATCH:

```
$ @ALI_DBTUNE_HOME:DBTUNE.COM
```

If executed in **BATCH**, DBTune expects the **ALI_RDB_IMPORT** logical to have been assigned prior to execution— you will not be prompted. To this end, a command file has been provided to allow assignment of the **ALI_RDB_IMPORT** logical. This command file—**DBTUNE_BATCH.COM**—can be found in the directory **ALI_DBTUNE_HOME**. After editing **DBTUNE_BATCH.COM**, DBTune can be invoked in batch with the command:

```
@$ @ALI_DBTUNE_HOME:DBTUNE_BATCH.COM
```

Following is a copy of the unedited batch command procedure:

```
$!------------------------------------------------------------------------
$!      DBTUNE_BATCH.COM
$!        - Command file to submit DBTune in BATCH mode...
$!
$!      Invoke this file with the command:
$!      $ @ALI_DBTUNE_HOME:DBTUNE_BATCH
$!------------------------------------------------------------------------
$!
$!      This command procedure will submit itself to batch.
$!
$       if p1 .eqs. "" .or. p1 .nes. "BATCH"
$           then
$!
$!      Change the /name="" qualifier to specify a different name for
$!      the job.
$!
$               cur_def = f$environment("DEFAULT")
$               vfl = f$verify(0)
$               set verify
$               submit-
$                   /log='cur_def'-
                    /noprint-
                    /name="DBTune in Batch"-
                    /parameters=("BATCH","''cur_def'") -
                    ALI_DBTUNE_HOME:DBTUNE_BATCH.COM
$               vfl = f$verify(vfl)
$               exit
$           endif
$!------------------------------------------------------------------------
$!      Change the following ASSIGN statement to point the database
$!      you wish to analyze and uncomment it by removing the "!" ...
$!
$! ASSIGN "disk1:[directory]database_name" ALI_RDB_IMPORT
$!
$!      Change the following SET PROC/NAME= to assign a different
$!      process name and uncomment it by removing the "!" ...
$!
$! SET PROC/NAME="DBTune in Batch"
$!
$!      Change the following SET DEFAULT to change the default
$!      directory where the report(s) will be generated. Otherwise,
$!      output will be generated the user's current directory at the
$!      time the file was submitted.
$!
$ SET DEFAULT 'p2'
$!------------------------------------------------------------------------
$ @ALI_DBTUNE_HOME:DBTUNE.COM
```

DBTune Parameters

Prior to executing DBTune, it is recommended that you review the default parameter settings provided with the tool. DBTune parameters allow you to control the Rdb transformation procedure that is generated. All of the parameters are pre-set to handle typical scenarios. You are encouraged to tailor these parameters to match your particular environment. This customization can be accomplished by editing the parameter file prior to invoking DBTune. DBTune then parses the parameter file and ensures that valid values have been selected. If a parameter value is found to be invalid, its default value is used.

The parameter file that DBTune will parse is pointed to by the logical ALI_DBTUNE_PARAMS. If this logical is not assigned prior to execution, DBTune will use the default parameter file that is provided (ALI_DBTUNE_HOME:DBTUNE_DEFAULT.PARAMS). To customize the parameter settings, you can do one of two things:

1. either edit the default parameter file directly

   or

2. copy the default parameter file to a new file with a different name and assign the logical **`ALI_DBTUNE_PARAMS`** to this new file.

If you are using DBTune for multiple databases, it is recommended to create a parameter file for each database. The parameter file contains environment information, such as available disk drives, along with the space that can be used for each database. Thus, using separate parameter files for each database will facilitate repeated use of DBTune with minimal setup time.

> **Note** It may be beneficial to carry this idea a step further and set up a subdirectory for each database that will hold all of the DBTune input and output files. This arrangement provides a great deal of easy-to-access documentation for each database.

Descriptions of each of DBTune's parameters and their possible values can be found later in this manual on page 130 under the section titled **DBTune Process - Step 2: Load Parameters**.

DBTune Keystrokes

The following keystrokes may be used when executing DBTune online using a VT220 (or higher) terminal interface:

[Help], [PF2]	Receive help for the current DBTune context or option.
[Esc], [PF3], [PF4]	Exit DBTune; if you are in a selection window to change a DBTune parameter, then function keys will simply return you to the main menu.
[Do]	Continue with the DBTune tuning process.
[Select]	Select a DBTune parameter to change for the current session.
[Ctrl] [W]	Refresh screen display.

After the database has been initially scanned and the screen displayed, a menu is presented on the bottom two lines of the display:

- **[DO]-**Create Tuning Scripts
- **[SELECT]-**Edit Parameters
- **[HELP]-**HELP
- **[ESC]-**Exit

If you are on a VT200+ terminal, these options can be executed with the designated function key. If you are on a PC or VT200- terminal, you can use the arrow keys to move around between the options. To select a function by means of the arrow keys, highlight the designated option and press **Return.**

If you wish to change a parameter setting, choosing the **Select** menu option will present a list of DBTune parameters. You can arrow between selections and press **Return** on the parameter to be changed. A window will then be presented in which to enter the new value. Online help is available at either the menu or in the change window. After completing the editing process, pressing **ESC** or **PF3** will return you to the main menu.

DBTune Process

The DBTune transformation process is automated to provide a simple method to achieve excellent Rdb performance. The process includes nine steps. The first eight steps occur automatically via the DBTune program and produce a command procedure that you manually execute during the ninth step to perform the Rdb transformation. The process can be executed online or in batch. The final transformation step can be invoked immediately upon its generation. It is recommended, however, to scan the **REVIEW_AND_GUIDE.REPORT** file created by DBTune and the DBTune log files that are generated during the first eight steps.

These documents will report any errors that may occur during the execution of DBTune as well as provide additional information to ensure successful execution of the transformation procedure. The ten steps of the DBTune process are:

0. Unload Statistics
1. Analyze Rdb
2. Load Parameters
3. Read Database Structure
4. Read Customized Analysis/Workload Data
5. Generate Performance Analysis Data File
6. Performance Analysis and Database Tuning
7. Generate Disk Utilization File

8. Generate Rdb Transformation Procedure

9. Transform Rdb

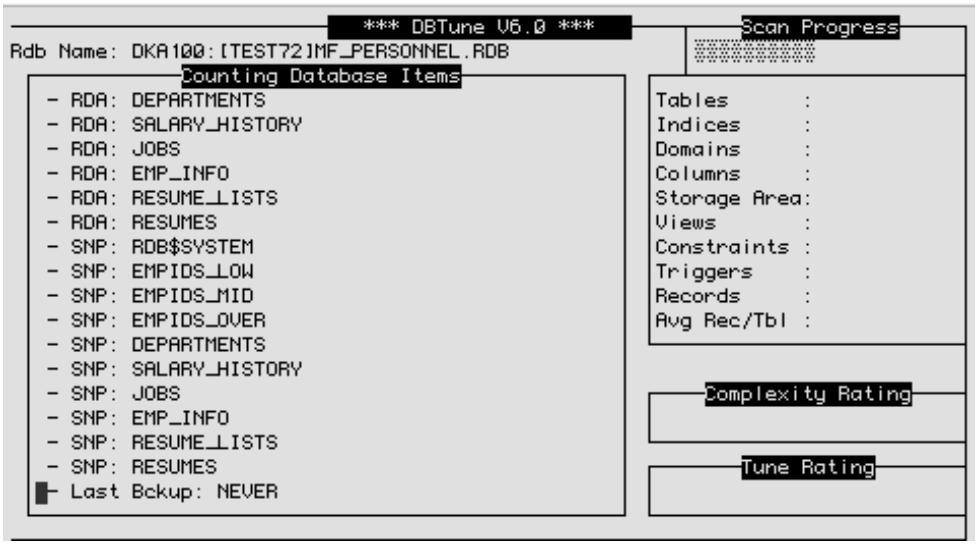
Note DBTune performs the first nine steps[0-8] ; you perform the last step[9].

Step 0: Unload Statistics

DBTune V6 collects all the database statistics for "your" database into its own internal database during the initial unload phase. The tool accesses "your" database in READ-ONLY mode and can be executed while other users are accessing the database. The first nine steps to the DBTune process then access the local DBTune database exclusively. The last step, which you execute manually, actually tunes the database and requires all users to be out of the database during its execution. Each of these steps is described in more detail in the following sections.

Step 1: Analyze Rdb

The database statistics for the logical and physical design are scanned. This information is summarized into Rdb component counts, a **COMPLEXITY** rating, and a **TUNE** rating. Additionally, a narrative is produced to explain the statistics and ratings in a format that emphasizes their performance impact.



This information is gathered during the initial phase of DBTune. The narrative report is written to the current default directory. The DBTune process can be stopped at this point so that the analysis can be reviewed before proceeding with the transformation process. The narrative analysis report is called **<<database_id>>_ANALYSIS.REPORT**, where **<<database_id>>** represents the first ten characters of the .RDB file name. For example, the report for the PERSONNEL database would be called:

PERSONNEL_ANALYSIS.REPORT

Step 2: Load Parameters

DBTune parameters allow you to control the Rdb transformation procedure that is generated. All of the following parameters are pre-set to handle typical scenarios. You are encouraged to tailor these parameters to match their particular environment. To customize parameter settings, you can do one of two things:

1. Edit the default or customized parameter file prior to executing DBTune or

2. Change the parameter online during DBTune execution by choosing the **[SELECT]-Edit Parameters** menu option. Not all parameters are available for editing online and any changes made will affect the current session only—the changes are NOT saved to the parameter file.

DBTune reads the parameter file and parses for exact matches on the parameter key word followed by an "**=**." Everything to the immediate right of the "**=**" is considered the parameter value.

Note If DBTune detects errors when parsing the parameter file, you will be given the opportunity to view a log of the errors found and to make corrections. If corrections are made, the parameter file is actually updated with the new value. You may also elect to continue without making corrections, allowing DBTune to substitute default values for erroneous parameters.

Following are descriptions of each of the (37) DBTune parameters, their effects and their possible values:

(1) STRATEGY

The **STRATEGY** parameter controls DBTune's handling of the database storage areas. There are three settings possible:

E Use **EXISTING** storage areas and leave them in their current physical locations. Existing areas may be resized. Empty storage areas are retained. Objects in RDB$SYSTEM (with the exception of partitioned objsts) are extracted however. Objects grouped together in an existing area remain grouped by default. Storage area names remain unchanged.

N Create all **NEW** storage areas, automatically distributing them to new locations across specified disks. Empty areas are dropped. Objects in RDB$SYSTEM are reassigned to new storage areas. Objects grouped together in an area are broken apart unless overridden by the DBA using the modpad file. Storage area names remain unchanged when possible, but new area names will also be generated as needed.

R A variation of "E" and "N"; use existing storage areas but **RELOCATE** them by automatically distributing to new locations across specified disks. Existing areas may be resized. Empty storage areas are retained. Objects in RDB$SYSTEM (with the exception of partitioned objsts) are extracted however. Objects grouped together in an existing area remain grouped. Storage area names remain unchanged.

Regardless of the STRATEGY setting, if tuning via SQL IMPORT, segmented strings (lists) will also be moved into new storage areas and will be distributed automatically across disks. Existing list area names will be retained if possible.

VALUES: **E** Use **EXISTING** storage areas

N Create all **NEW** storage areas

R Use existing, but **RELOCATE**

DEFAULT: **N**

(2) TUNE_TECHNIQUE

The **TUNE_TECHNIQUE** parameter determines the method used to tune a database. An entire database can be tuned all at once by using a SQL EXPORT/IMPORT, or parts of a database can be tuned by using RMU UNLOAD/LOADs. If RMU UNLOAD/LOADs are chosen to tune a database, you can specify particular tables and/or indexes to be tuned while leaving other tables/indexes alone. In addition, you can specify a time limit (in minutes) for the UNLOAD/LOAD process in which only those tables that provide the most performance benefit will be tuned. This time limit can be specified via the DBTune parameter LOAD_TIME_LIM.

Note If the database being tuned is a single file database, **RMU UNLOAD/LOAD**s will NOT be used, regardless of the setting for the **TUNE_TECHNIQUE** parameter. **SQL EXPORT/IMPORT**s will be used on all single file databases. If **RMU UNLOAD/LOAD**s are chosen as the tuning technique, neither the **RDB$SYSTEM** storage area nor any **LIST** (segmented string) areas will be tuned. Thus, the only way to tune tables/indexes stored in **RDB$SYSTEM** or to tune segmented strings is to use **SQL EXPORT/IMPORT**.

Note If you choose to use **RMU/UNLOADS** and **RMU/LOADS** to tun selected tables of a database, items which are defined on those tables (views, constraints, triggers, comments) must be dropped and then re-created when the tables are dropped and re-created. If any views, constraints, or triggers were previously defined with RDO, the tuning scripts may fail when the items are re-created using SQL statement because DBTune uses the original definition of the item to re-create it. If database items were formerly created using RDO, you should review the SQL scripts generated by DBTune BEFORE executing the **MAIN_DRIVER** command file to ensure that no syntactical violations exist for views, constraints, and/or triggers.

Warning The **RMU** setting will create scripts that require an **INTERACTIVE** or **DEVELOPMENT** license for Rdb/VMS and will fail if you only have a **RUNTIME** license! If the scripts that DBTune generates are executed on a system that only has the **RUNTIME** license for Rdb/VMS, you MUST set the **TUNE_TECHNIQUE** to **SQL**.

VALUES:	**SQL**	Use SQL EXPORT/IMPORT
	RMU	Use RMU UNLOAD/LOADs
DEFAULT:	**SQL**	

(3) DBDISKS

The **DBDISKS** parameter determines the number of logical storage devices that are to be used for the database being tuned. "New" storage areas (those that reside in RDB$SYSTEM, those that contain segmented strings, or those being relocated because STRATEGY = N or R) will be spread over the various disks specified (via the **DBDISKnn** parameters below) to reduce the I/O load on any particular disk. The more disk devices specified, the more a database can benefit from the storage area distribution.

VALUES: 1 to 999 logical devices

DEFAULT: 1

(4) DBDISKnnn

The **DBDISKnnn** parameters are used in conjunction with the DBDISKS parameter above to specify actual disks and directories to be used for spreading storage areas. In addition, the available blocks can be specified for a particular disk as well as the type of storage area files you wish to be placed on that disk. The following naming convention should be used when specifying these parameters: "DBDISKnnn" where "nnn" is "001", "002", "003" . . . up to "999". Both available blocks and storage file qualifiers can be specified after the disk and directory specification using "/"s to separate the values. For example, if the previous parameter was specified as "DBDISKS=4", the actual disks and directories to be used could be specified in the following manner:

```
DBDISK001=DISK1:[MYDATA.RDB]/25000/SYSRDB/SYSRDA/
DBDISK002=DISK2:[MORE_DATA]/150000/TBLRDA/IDXRDA/
DBDISK003=DISK3:[EVEN_MORE]/65000/TBLSNP/IDXSNP/SYSSNP/
DBDISK004=DISK4:[WHOA]
```

where 25000, 150000 and 65000 are the maximum blocks of free space DBTune is allowed to use on the first three disks, respectively. For the fourth disk, DBTune will attempt to determine the available blocks on the physical disk and use that value. If unable to determine the available blocks, DBTune will assume the fourth disk has "all available space" as accessible.

For this example, any DBDISKnnn parameter greater than DBDISK004 (e.g., DBDISK005) would be ignored because the parameter setting of "DBDISKS=4" limits the number of DBDISKnnn parameters that will be used to four.

By default, the first such DBDISKnnn parameter (DBDISK001) is created for the user and left blank. You can add more DBDISKnnn parameter lines as required by the DBDISKS parameter setting.

Valid storage file qualifiers are as follows:

SYSRDB .RDB file for the database (root area)

SYSRDA .RDA file for the database system area (RDB$SYSTEM)

SYSSNP .SNP file for the database system area (RDB$SYSTEM)

TBLRDA .RDA files for TABLE storage areas

TBLSNP .SNP files for TABLE storage areas

IDXRDA .RDA files for INDEX storage areas

IDXSNP .SNP files for INDEX storage areas

If NO file qualifiers exist on a DBDISKnnn line, then ANY type of storage file can be stored there. But, if a file qualifier is specified for a DBDISKnnn parameter, then ONLY those types of storage files can be stored in the specified location. A typical use of this feature would be to specify a particular location for the database root file(s) or to force all snapshot files to be placed on a particular disk, etc. The storage file qualifiers shown for the previous four disks specify the following:

DBDISK001 allow ONLY the .RDB and .RDA files for the database system area (root) to be stored here

DBDISK002 allow ONLY .RDA files for table and index storage areas to be stored here

DBDISK003 allow ONLY .SNP files for tables, indexes, and the database system area to be stored here

DBDISK004 allow ANY type of file to be stored here

Following is an example of how to prevent all index files (.RDAs and .SNPs) from being stored on a particular disk while allowing any other type of file to be stored there:

```
DBDISK001=DISK1:[RDB]/SYSRDB/SYSRDA/SYSSNP/TBLRDA/TBLSNP/
```

If the database requires more space than is assigned to the DBDISKnnn parameters or storage file qualifiers become too restrictive (causing DBTune to run out of locations to place storage area files), DBTune will create an initial "OVERFLOW" disk whose default location is the current location of the database .RDB file. If additional overflow space is needed, DBTune will place the files in ALI_DEFAULT_DIR (if the logical is defined), or the directory from which the DBTune is beiing run (ie SYS$LOGIN for batch jobs). You can override this default location in the Disk Utilization file during DBTune processing if the EDIT_FILES parameter is set to **Y**.

Note DBTune assumes that the fastest disk device will be listed first (DBDISK001), the second fastest disk second (DBDISK002), etc.

Important DBTune V6 considers each of the **DBDISKnnn** parameters to be a separate device, even though you may assign them all to the same physical disk. Thus, if the following assignments are made:

DBDISK001=DISK1:[MYDATA.RDB] / 10000 /
DBDISK002=DISK1:[MYDATA.SNAP] / 10000 /

DBTune will limit the allocated files in each DBDISKxxx location to the amount specified. DBTune will utilize a max of 20000 blocks on DISK1 because even though it considers DBDISK001 and DBDISK002 to be separate devices, they are in fact different directories on the same physical device. DBTune will correctly identify, and monitor, that a total of 20000 blocks have been requested for the physical disk (DISK1 to prevent overallocation. Appropriate warnings are placed in the DBTune log file when space is becoming critical (< 100,000 free blocks on the physical device DISK1).

DBTune will take into account the space which is to be "freed" during the tuning process (as objects are dropped or moved), and which will then be available for reuse later in the tuning process when area files are re-created.

Caution If after-image journaling is enabled for your database, it is highly recommended NOT to specify the disk on which the AIJ file resides as a value for one of the DBDISKnnn parameters.

(5) EDIT_FILES

The **EDIT_FILES** parameter controls whether or not the DBTune process will pause and allow you to edit the Performance Analysis Data (PAD) file and later, the Disk Utilization file. DBTune automatically generates these files, but you can edit them during the DBTune process to further tailor the information to affect tuning and storage area spreading.

Note If you plan to tailor the **PAD** file and use DBTune to maintain a database, it is recommended to create a ModPAD file rather than repeatedly editing the **PAD** file online during DBTune execution. The ModPAD file facilitates continuous maintenance of the database by "seeding" the online PAD file with values each time DBTune is executed. To repeatedly use the same ModPAD file for a database, the **MODPAD_FILE** parameter below can be utilized.

VALUES:	**N**	Do NOT Edit Performance Analysis Data File during DBTune session

	Y	Edit Performance Analysis Data File during DBTune session

DEFAULT: **N**

(6) ALI_EDITOR

The **ALLI_EDITOR** parameter is used to specify the editor to be invoked if the **EDIT_FILES** parameter is set to **Y**. If no editor is specified, then **EDIT/EDT** will be used.

EXAMPLES: EDIT/EDT, EDIT/TPU, etc.

DEFAULT: EDIT/EDT

(7) MODPAD_FILE

The **MODPAD_FILE** parameter, if not blank, indicates that DBTune will generate its online **Performance Analysis Data** (PAD) file based on the specified ModPAD file. The ModPAD file is used to "seed" the online **PAD** file, allowing you to consistently use the same performance data to facilitate continuous maintenance of a particular database (in much the same way the **MODPARAMS.DAT** file is used to seed the **PARAMS.DAT** file for the VMS AUTOGEN utility). If the file specified for **MODPAD_FILE** does not yet exist, then DBTune issues a warning message, but attempts to generate a brand new ModPAD file with the name and location specified by this parameter. However, if the file specified for **MODPAD_FILE** does exist, it is used to pre-set values in the online **PAD** file. In addition, if there have been any changes in the database (tables/indexes added or dropped), the specified ModPAD file will be updated with the new items.

EXAMPLE: MODPAD_FILE=PERSONNEL.MODPAD

Important If an "@" character is entered in a changeable column in the ModPAD file, DBTune will interpret the "@" character to mean "replace with current database value." Thus, if an "@" character is entered in the Number of Recs column for a particular table, DBTune will convert the "@" into the actual cardinality that exists for that table in the database. If a value in a column is a "hard-coded" value (e.g., 25000 records), DBTune will keep that value and NOT override it with the actual database value. Thus, the ModPAD file can contain both variable and static information to be used for the database tuning. See the next page for an example.

Note Changes made to the online **PAD** file during execution of DBTune are NOT saved to the ModPAD file!

Note Any **DBDISK** information entered into the ModPAD file is ignored.

Note The filename entered for the **MODPAD** parameter CANNOT have the file extension **.PAD** because this may conflict with creation of the associated **PAD** file. To be safe, name ModPAD files with the extension **.MODPAD** or **.MOD**.

The example below shows a ModPAD file with explanations of the various symbols and syntax used:

```
! ModPAD (used to modify online PAD file)
! TABLE Section:
! ==============
! "Grw %" column   : Table Growth Percentage (values: 0...999)
! "Acc Bia" column : Access Bias (values: 0..100; 0=100% Write, 100=100%Read
! "Act Lvl" column : Table Activity Level (values: 1..9 with 1=Low,9=Hi)
! "Snp %" column   : Snapshot Percentage (values: 0...999)
! "Tun Tbl" column : Tune Table? (values: Y-Yes, N-No; only for RMU/LOADs)
! "Ena Cmp" column : Enable Compression? (values: Y-Yes, N-No)
!                                    Number    Grw Acc Act Snp Tun Ena
! Database Table Name                of Recs    %  Bia Lvl  %  Tbl Cmp
! ---------------------------------- -----*------ -*- -*- -*- -*- -*- -*-
[ CANDIDATES ]                     /     @      /850/ 50/ 5 / 25/ Y / @ /
[ COLLEGES ]                       /     @      / @ / 30/ 1 / 75/ N / @ /
[ DEGREES ]                        /       10000/  0/ 70/ 9 / @ / Y / @ /
[ DEPARTMENTS ]                    /     @      / 30/ @ / 7 / 10/ Y / @ /
```

Explanation of the above TABLE entries:

Row 1: Use the values that are "hard-coded", but replace "Number of Recs" with the actual cardinality for this table.

Row 2: Replace “Number of Recs” and replace “Growth %” with the value specified for the DBTune “GROWTH” parameter. Do NOT tune this table if **TUNE_TECHNIQUE = RMU**.

Row 3: Use “10000” instead of using the actual cardinality for the table and replace “Snapshot %” with the value specified for the DBTune “SNP_PERC” parameter.

Row 4: Replace “Number of Recs” and replace “Access Bias” with the value specified for the DBTune “BIAS” parameter.

```
! INDEX Section:
! ==============
! 'Idx Typ' column : Index Type (values: S-Sorted, R-Ranked,
!                                        H-Scattered, O-Ordered)
! "Act Lvl" column : Index Activity Level (values: 1..9 with 1=Low,9=Hi)
! "Avg Dups" column: Average Duplicates for an index (values: 0...9999)
! "Key Nod" column : Index Key Values Per Node (values: 3...999)
! "Fil %" column   : Index Node Fill Percentage (values: 33...100)
! "Snp %" column   : Snapshot Percentage (values: 0 to 999)
! "Tun Idx" column : Tune Index? (values: Y-Yes, N-No; only for RMU/LOADs)
!                                  Idx Act Avg  Key Fil Snp Tun
! Database Index Name              Typ Lvl Dups Nod  %   %  Idx
! -------------------------------- -*- -*- --*- -*- -*- -*- -*-
{ DEG_COLLEGE_CODE }              / H / 2 /  @ / @ / 90/ 25/ Y /
{ DEG_EMP_ID }                    / S / 7 /  @ / @ / @ /150/ Y /
{ EMP_EMPLOYEE_ID }               / H / 3 /  @ / @ / 90/ @ / Y /
{ EMP_LAST_NAME }                 / S / 8 / 10 / 17/ 90/400/ Y /
```

Explanation of the above INDEX entries:

Row 1: Use the values that are “hard-coded”, but replace “Avg Dups” with the current average number of duplicates for this index and re-calculate a new value for “Key Values Per Node”.

Row 2: Replace “Avg Dups”, calculate “Key Values Per Node”, and replace “Node Fill %” with the value specified for the DBTune “FILL” parameter.

Row 3: Replace “Avg Dups”, calculate “Key Values Per Node”, and replace “Snapshot %” with the value specified for the DBTune “SNP_PERC” parameter.

Row 4: Use all “hard-coded” values, do not substitute or re-calculate any value for this index.

```
! CLUSTER Section:
! ================
[DEGREES] & { DEG_COLLEGE_CODE }
[DEGREES] = { DEG_COLLEGE_CODE }
! CONTRA Section:
! ===============
[ CANDIDATES ]~[ COLLEGES ]
```

(8) DYNAMIC_WORKLOAD_FILE

The **DYNAMIC_WORKLOAD_FILE** parameter, if not blank, indicates that DBTune is to use the Dynamic Workload data file produced by a database monitor process. This file should contain activity weightings for tables and/or indexes and can be created automatically via DBXAct, manually by the user, or by a third-party tool. The information contained is the result of a transaction analysis on a database. It weights the tables and indexes on a scale of 1 to 9, where 9 indicates the highest activity and 1 indicates the lowest. This information is merged with the Performance Analysis Data file during execution of DBTune. The Performance Analysis considers the activity information when optimizing the distribution of storage areas across the available disks. If a file specification is entered for this parameter, the file will be validated for existence and access privilege. The file is expected to be an RMS sequential file (CONTROL=CARRIAGE_RETURN, VARIABLE LENGTH) and if the file is unable to be accessed, DBTune will ignore it.

EXAMPLE: DYNAMIC_WORKLOAD_FILE=
PERSONNEL.DYN_ACT

> **Note** **Activity Level** values in the **DYNAMIC_WORKLOAD_FILE** override **Activity Level** values specified in the **MODPAD_FILE**.

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! Dynamic Database Activity
!
! For Database: MY_DATABASE.RDB
!
*SOURCE=MANUAL
*BEGAN = 07-Feb-2002 11:43:27
*ENDED = 14-Feb-2002 15:21:56
!
!
!                                 Activity
!                                   Level    Growth%
! Item Name: [TABLE] or {INDEX}  (9-HI,1-LO) (0-999)  Access Bias
```

```
!------------------------------ ----------- ------- ------------
[EMPLOYEE_TABLE]              /     9     /   10  /     50     /
[INVOICE_TBL]                 /     7     /  117  /     25     /
{EMP_NUM_IDX}                 /     6     /
{INV_NUM_KEY}                 /     2     /
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

(9) SQL_DIR

The **SQL_DIR** parameter assigns the disk and directory where DBTune will create the SQL scripts and DCL command procedures to tune the database.

VALUES:	CURRENT	(use current default directory)
	disk:[dir]	(specify some other directory)
DEFAULT:	CURRENT	

(10) BACKUP_DIR

The **BACKUP_DIR** parameter assigns a disk and directory that will contain the RMU/backup of the database. You can prevent the backup from occurring by specifying NONE. It is strongly recommended to allow DBTune to perform a backup prior to tuning the database so that recovery is possible in the unlikely event of system failure or some other event that prevents the tuning process from completing successfully. It is recommended you use a scratch disk with abundant free blocks to ensure the tuning process has sufficient space to store the existing database. The **REVIEW_AND_GUIDE.REPORT** created by DBTune contains information on the estimated storage requirements for the backup. Your system manager may be able to provide a more precise storage requirement based on previous backups.

VALUES:	NONE	(no backup performed)
	CURRENT	(use current default directory)
	disk:[dir]	(specify some other directory)
DEFAULT:	CURRENT	

Note You can specify a tape device for this parameter. However, in order for DBTune to validate the parameter, it will attempt to create a file on the specified device. For this to occur, a tape must already be loaded and mounted into the tape drive PRIOR to running DBTune. The tape should already have been INITIALIZEed and should be MOUNTed as a files device (without the /FOREIGN qualifier).

(11) EXPORT_UNLOAD_DIR

The **EXPORT_UNLOAD_DIR** parameter assigns a disk and directory that will contain the database **EXPORT** file if performing a SQL **EXPORT/IMPORT** or the **UNLOAD** data files if performing **RMU/UNLOADs** and **LOAD**s. The export/unload files are temporary files used to transform the database into the new design. You should you use a scratch disk with abundant free blocks because the export/unload files are an uncompressed copy of the data and may be larger than the actual database.

The **REVIEW_AND_GUIDE.REPORT** file created by DBTune contains information on the estimated storage requirements for the export/unload files. Your system manager may be able to provide a more precise storage requirement based on previous exports/unloads.

VALUES:	CURRENT	(use current default directory)
	disk:[dir]	(use some other directory)
DEFAULT:	CURRENT	

Note You can specify a tape device for this parameter. However, in order for DBTune to validate the parameter, it will attempt to create a file on the specified device. For this to occur, a tape must already be loaded and mounted into the tape drive PRIOR to running DBTune. The tape should already have been INITIALIZEed and should be MOUNTed as a files device (without the /FOREIGN qualifier).

(12) RUJ_DIR

The **RUJ_DIR** parameter assigns a disk and directory that will contain the **RUJ** (Recovery Unit Journal) file for the **MAIN_DRIVER** process that actually tunes the database. The **RUJ** file is a temporary file used to transform the database into the new design. You should use a scratch disk with abundant free blocks as the **RUJ** file must store a copy of each table row that is being loaded/unloaded or altered and can grow quite large. If the disk that contains the **RUJ** file runs out of free blocks during execution of the tuning procedure, the procedure will fail. To approximate how much disk space is required for the **RUJ** file, estimate the size of the largest table in the database (in blocks) and multiply by two. At a minimum, there should be at least 20000 blocks of free space on the disk assigned to the **RUJ_DIR** parameter.

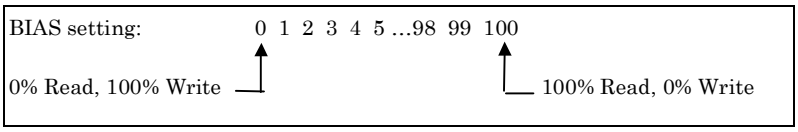
Note If a VMS logical is used in the **RUJ_DIR** specification, it MUST be assigned at the **SYSTEM** level to ensure proper access by the database. If a non-SYSTEM logical is used, the SQL tuning scripts may fail.

VALUES:	CURRENT	(use current default directory)
	disk:[dir]	(use some other directory)
DEFAULT:	CURRENT	

(13) BIAS

The **BIAS** parameter controls whether the Performance Analysis Data file is initially loaded with READ or WRITE biased tables and indexes. READ bias will create larger NODE sizes and PAGE sizes for those storage areas that contain a sorted index, reducing the number of Direct I/Os. WRITE bias will create smaller NODE and PAGE sizes, reducing lock contention, but increasing the number of Direct I/Os. The BIAS setting has a global effect on the database, but you can change individual table settings in the **PAD** and/or **ModPAD** files.

The **BIAS** setting represents a sliding scale with 100 percent **WRITE** bias on one end and 100 percent **READ** bias on the other end. For example:



If a table is accessed 50 percent of the time with **READ ONLY** transactions, its rating would be 50. If a table is accessed 80 percent of the time with **READ WRITE** (update/insert/delete) transactions, its rating would be 20 (it is **READ ONLY** 20 percent of the time).

VALUES: 0 to 100

DEFAULT: 50 (50% Read, 50% Write)

(14) FILL

The **FILL** parameter value is used to control the initial placement of index records in each index node (for **SORTED** indexes only). Generally, for **READ** biased (retrieval-intensive) indexes, a **FILL** factor greater than or equal to 70 is recommended. For **WRITE** biased (update-intensive) indexes, a **FILL** factor of less than or equal to 70 is recommended. This **FILL** factor is a global setting for the database, but you can change this value for individual indexes in the **PAD** and/or **ModPAD** files.

VALUES: 33 to 100

DEFAULT: 90

(15) GROWTH

The **GROWTH** parameter sets the percentage of additional space that will be allocated to existing tables and indexes. For example, a GROWTH parameter value of 30 would cause an additional 30 percent to be allocated to the existing cardinality of each table (i.e., allocation would be calculated for 130 records if the table currently contained 100 records). This parameter is a global setting that affects all tables and indexes in the database. You can override this setting for individual tables, however, via the PAD and/or ModPAD files.

VALUES: 0 to 999

DEFAULT: 10 [recommended to be 30 if ample disk space available]

(16) SNP_PERC

The **SNP_PERC** parameter controls how much space is allocated to the snapshot file for each storage area. **SNP_PERC** represents a percentage of the allocation for the associated **.RDA** file. For example, if an **.RDA** file is allocated 200 pages and **SNP_PERC** is set to 25, then the .**SNP** file will be allocated 50 pages (25 percent of 200). This parameter is a global setting that affects all tables and indexes in the database. You can override this setting for individual tables/indexes, however, via the **PAD** and/or **ModPAD** files.

VALUES: 0 to 500

DEFAULT: 5 [recommended to be 25 if ample disk space available]

(17) MIN_PAGE_SIZE

The **MIN_PAGE_SIZE** parameter controls the minimum page size (in blocks) for database storage areas that are tuned via DBTune. The value may range from 1 to 32 blocks. For certain READ-intensive applications, larger page sizes may improve direct I/O performance. For WRITE-intensive applications, however, larger page sizes may increase locking contention. The default **MIN_PAGE_SIZE** will produce optimal performance for applications that are divided equally between both READs and WRITEs. If **TUNE_TECHNIQUE = RMU**, this parameter cannot exceed the existing buffer size of the database.

VALUES: 1 to 32

DEFAULT: 1

(18) MAX_PAGE_SIZE

The **MAX_PAGE_SIZE** parameter controls the maximum page size (in blocks) for database storage areas that are tuned via DBTune. The value may range from 2 to 32 blocks, but cannot be less than **MIN_PAGE_SIZE** and cannot exceed **MAX_BUFFER_SIZE**. For certain READ-intensive applications, larger page sizes may improve direct I/O performance. For WRITE-intensive applications, however, larger page sizes may increase locking contention. The default **MAX_PAGE_SIZE** will produce optimal performance for applications that are divided equally between both READs and WRITEs.

VALUES: 2 to 32

DEFAULT: 32

(19) MIN_BUFFER_SIZE

The **MIN_BUFFER_SIZE** parameter controls the minimum buffer size for a database tuned via DBTune. Buffer size is only calculated for a database when **TUNE_TECHNIQUE = SQL**. The value may range from 3 to 64 blocks. Buffer size impacts the size of process working sets required for database users. A large buffer size or a large number of buffers may require larger working sets, while a small buffer size or a small number of buffers may result in poor direct I/O performance for certain applications. DBTune will automatically calculate a buffer size for a database based on the sizes of its tuned storage areas. The **REVIEW_AND_GUIDE.REPORT** created by DBTune provides information on resources impacted by the resulting buffer usage.

VALUES: 3 to 64

DEFAULT: 6

(20) MAX_BUFFER_SIZE

The **MAX_BUFFER_SIZE** parameter controls the maximum buffer size for a database tuned via DBTune. Buffer size is only calculated for a database when **TUNE_TECHNIQUE = SQL**. The value may range from 3 to 64 blocks, but cannot be less than **MIN_BUFFER_SIZE** or **MAX_PAGE_SIZE**. Buffer size impacts the size of process working sets required for database users. A large buffer size or a large number of buffers may require larger working sets, while a small buffer size or a small number of buffers may result in poor direct I/O performance for certain applications. DBTune will automatically calculate a buffer size for a database based on the sizes of its tuned storage areas. The **REVIEW_AND_GUIDE.REPORT** provides information on resources impacted by the resulting buffer usage.

VALUES: 3 to 64

DEFAULT: 64

(21) MIN_BUFFERS

The **MIN_BUFFERS** parameter controls the minimum number of buffers assigned to the database; the value may range from 0 to 250. DBTune will start with this number of buffers and add to it based on the resulting physical design. Buffers can reduce disk I/O by utilizing data retrieved via previous I/Os but they also require additional memory from the process working set. DBTune's **REVIEW_AND_GUIDE.REPORT** provides information on resources impacted by the resulting buffer usage.

VALUES: 0 to 250

DEFAULT: 20

(22) MAX_BUFFERS

The **MAX_BUFFERS** parameter controls the maximum number of buffers assigned to the database; the value may range from 20 to 1000. DBTune ensures that **MAX_BUFFERS** is greater than or equal to **MIN_BUFFERS**. DBTune's **REVIEW_AND_GUIDE.REPORT** provides information on resources impacted by the resulting buffer usage.

VALUES: 20 to 1000

DEFAULT: 100

(23) SYS_MEM_PAGES

The **SYS_MEM_PAGES** parameter controls the number of physical memory pages to be used by a database for global buffers. Global buffers can improve performance by reducing I/O operations and by better utilizing memory. To see how many memory pages are available or "free" on a system, type the following command at the DCL prompt: **`$ SHOW MEM/PHYS`**. If the database being tuned can be accessed from more than one node or system, the value entered for this parameter should not exceed the LEAST number of free pages available on the candidate systems. For example, if MY_DATABASE can be accessed from Node A (20000 free pages), Node B (15000 free pages), and Node C (50000 free pages), **SYS_MEM_PAGES** should be set no higher than 15000. If this parameter is set to 0, the existing global buffer settings for the database will be maintained. If this parameter is set to a value greater than 0, global buffers will be enabled for the database and will be set to maximize usage of the specified system memory pages. If the value entered for this parameter is insufficient for a minimum number of buffers (5) to be allocated to the maximum number of database users, an error message will be given and the parameter will be reset to 0. DBTune's **REVIEW_AND_GUIDE.REPORT** provides information on resources impacted by any resulting buffer usage.

Note Do NOT assign ALL available pages of memory to the **SYS_MEM_PAGES** variable. It is safest to leave some pages in reserve in the event of a system emergency or an abnormally heavy user load where extra pages may be needed.

VALUES: 0 to 999999999999

DEFAULT: 0

(24) MAX_DB_USERS

The **MAX_DB_USERS** parameter controls the maximum number of users allowed to access the database at one time. Each process (online and batch) that attaches to the database is considered a "user." The number of database users, in conjunction with the **SYS_MEM_PAGES** parameter, affects the number of global buffers allocated to each user. If this parameter is set to 0, the existing number of users for the database will be maintained. If this parameter is set to a value greater than 0, it will override the existing database setting.

VALUES: 0 to 2032

DEFAULT: 0

(25) STOR_AREA_SPREAD

The **STOR_AREA_SPREAD** parameter controls what value will be used to spread database storage areas over the disks specified above. Each storage area is weighted according to this parameter. The weighting is used along with **CONTRA** and **CLUSTER** information to place storage areas on available disks to enable maximum I/O throughput and reduce I/O bottlenecks.

VALUES:	**A**	Spread areas based on **ACTIVITY**
	V	Spread areas based on **VOLUME**
	B	Spread areas based on a factor of **BOTH** Activity and Volume

DEFAULT: **B**

(26) LOGICALS

The **LOGICALS** parameter determines if VMS logicals will be generated for "new" and "relocated" storage area files (RDA and SNP). The logicals will be named using the following format:

<<db_id>>_<<storage_area_name>>_RDA for .RDA files and
<<db_id>>_<<storage_area_name>>_SNP

for .SNP files

where **<<db_id>>** is the first ten characters of the database name.

If logicals are not used, the new storage areas will be created in the physical location specified for the DBDISK to which they were assigned in the Disk Utilization file.

If specified, the logicals will be assigned so as to distribute the storage areas according to the Disk Utilization file.

Note If the **STRATEGY** parameter is set to **E** (Existing), then the existing logicals and/or file specification for each storage area is used. Any logicals used for these **EXISTING** specifications are assigned 'unchanged' as part of the DBTune **Transform** process.However, since new list areas may be created, it is advisable to use (or at least review) these "new" logical definitions.

If the **STRATEGY** parameter is set to **R** (Relocate), then the existing logicals , and new file specifications for each storage area are used. Any logicals used for these **RELOCATED** specifications are automatically assigned inside of the DBTune **Transform** process. These new logical definitions will have to replace previous definitions that may have been used.

VALUES: **Y** (**YES,** use logicals)
N (**NO**, do NOT use logicals)

DEFAULT: **N**

(27) LOGICAL_TYPE

The **LOGICAL_TYPE** parameter indicates what type of logical is to be created if the **LOGICALS** parameter is set to **Y.**

VALUES: **SYSTEM**
PROCESS
GROUP
JOB
TABLE = logical_name_table
where logical_name_table
and LNM$FILE_DEV
have been pre-defined by the user

DEFAULT: **PROCESS** (but **SYSTEM** is recommended)

> **Note** If logicals are specified, it is highly recommended to use **SYSTEM** logicals for production Rdb databases. Be sure to place any changed definitions in the proper startup file for the system in question.

(28) CONCEAL_LOGS

The **CONCEAL_LOGS** parameter indicates whether the logicals that are generated are **CONCEALED** and **ROOTED**.

> **Note** If **CONCEALED** logicals are not used, then RDB translates the logicals to their physical device. In this case, later reassignment of storage areas may require you to execute **RMU/MOVE** manually for each storage area with a non-concealed logical. **CONCEALED** logicals can only be used if they are assigned to rooted directory specifications.

VALUES: **Y** (**YES**, use CONCEALED, ROOTED logicals)
N (**NO**, do NOT use CONCEALED, ROOTED logicals)

DEFAULT: **N**

(29) LOAD_TIME_LIM

The **LOAD_TIME_LIM** parameter determines the number of minutes allowed for performing **RMU UNLOAD/LOAD**s on a database. If **LOAD_TIME_LIM** equals zero (0), DBTune assumes it has "unlimited" time to perform the **UNLOAD/LOAD**s. If **LOAD_TIME_LIM** is greater than zero, DBTune will determine which tables/indexes will provide the most performance benefit when tuned and then tune as many as possible in the number of minutes specified. A common use for this parameter is to set it for the specific window of time that your system can be down and then see how much of the database can be tuned during that period. You can "pre-select" which tables/indexes will be considered for tuning via a **MODPAD_FILE**. DBTune will ignore all tables whose **Tbl Tun** value is **N** and all indexes whose "**Idx Tun**" value is **N** in the **MODPAD_FILE**. Only those tables/indexes whose "Tune" value is **Y** will be considered for tuning. If the **LOAD_TIME_LIM** parameter is set to a value greater than zero, a Cost/Benefit analysis for items chosen to be tuned within the specified time limit will be printed in the **dbname_DBTUNE.LOG** output file.

Note The **LOAD_TIME_LIM** parameter is used only if **TUNE_TECHNIQUE = RMU**.

Note If you wish to see how long it will take to tune the entire database using **RMU/LOADS**, set **LOAD_TIME_LIM** to 30000, run DBTune, and then read the **<<DB_ID>>_DBTUNE.LOG** file that DBTune generates.

VALUES: 0 (unlimited) to 30000 minutes

DEFAULT: 0

(30) MACHINE_VUPS

The **MACHINE_VUPS** parameter is used ONLY when performing **RMU UNLOADs/LOAD**s to estimate the total tuning time required. This parameter should represent the **VUP**s rating for a single processor on the system from which tuning will occur. For example, if a database will be tuned on Node A (a VAXstation 3100 with a VUPs rating of 2.5), tuning may take longer than if it occurs on Node B (a MicroVAX 4000-200 with a VUPs rating of 5.0). If a system has multiple processors (i.e., VAX 6440 has four processors of 6 VUPs each for a total of 24 VUPs), it is assumed that the tuning process can only take advantage of a single processor. So, in this case, the **MACHINE_VUPS** parameter would be set to 6 not 24. The more VUPs a processor has, the quicker the tuning process will complete. Disk I/O rates and disk fragmentation can also affect the time required to tune a database; however, these variables are considered uniform at the present time and are not allowed to be specified.

VALUES: 1.0 to 999.9 VUPs

DEFAULT: 2.5

(31) TABLE_COMMIT

The **TABLE_COMMIT** parameter is used ONLY when performing **RMU UNLOADs/LOAD**s. One of the intermediate steps involved in tuning via **RMU/LOAD**s is to delete (drop) the tuned tables from the database and then add them back. When a table is dropped from the database, the **RUJ** file can grow quite large. To keep the size of the **RUJ** file down to a manageable size, a **SQL COMMIT** can be performed after every **DROP TABLE**. This has the disadvantage, however, of not allowing a complete rollback if a later table drop fails. On the other hand, if all the tables are dropped and then a **COMMIT** is performed once at the end, a complete rollback can be performed but the **RUJ** file may grow so large that it runs out of disk space, causing the tuning scripts to fail. Thus, if the tables being tuned are very large (e.g., have over 500000 records) and/or disk space is limited, it might be a good idea to **COMMIT** after EVERY table is dropped. However, if tables are not unusually large, or if disk space is abundant, or if you want all changes to be rolled back if an error occurs, drop all the tables and perfom a **COMMIT** once at the end.

VALUES: **N** **NO**, do NOT commit after every table drop (results in large RUJ)

	Y	**YES**, commit after EVERY table drop (unable to rollback fully in case of error)
DEFAULT:	**Y**	

(32) SAVE_COMMENTS

The **SAVE_COMMENTS** parameter is used ONLY when performing **RMU UNLOADs/LOAD**s. When tables are temporarily dropped and then added back during the process of tuning with **UNLOAD**s and **LOAD**s, all items that refer to that table are also dropped and then added back. This is also true for **COMMENTS** (table, table.column, and index comments). The process of checking for and restoring comments for every table that is dropped can be a lengthy one. If you do not use comments on your database or do not care to have them restored, you can save time by skipping comments altogether. The presence or absence of comments does not affect the performance of the database. This parameter is used only to lessen DBTune's processing time for **UNLOAD**s and **LOAD**s.

VALUES:	**N**	**NO**, do NOT try to restore comments that may have been dropped
	Y	**YES**, restore all comments that may have been dropped
DEFAULT:	**Y**	

(33) SA_MIN_CARD

The **SA_MIN_CARD** parameter is used to set the minimum cardinality threshold that will cause DBTune to create a separate storage area for a table and each of its indexes. Tables whose cardinality falls below this threshold will be grouped together in a single storage area named by the **SMALL_TABLE** parameter. Indexes for those tables will similarly be clustered together in either **SMALL_SORTED** or **SMALL_HASHED**.

VALUES:	0 to 100000
DEFAULT:	100 (not recommended to exceed 1000)

(34) SMALL_TABLE

The **SMALL_TABLE** parameter is used to set the common storage area name for tables that have a record count less than the SA_MIN_CARD parameter. **IT IS HIGHLY RECOMMENDED THAT THE USER PRECEDE THE "SMALL_TABLE" NAME WITH THE NAME OF THE DATABASE IF STORING MORE THAN ONE DATABASE IN THE SAME DIRECTORY!** For example, if storing the INVOICE database and the ORDERS database in the same directory, the values for this parameter might be "INVOICE_SMALL_TABLE" or "ORDERS_SMALL_TABLE", depending on which database was being tuned during a session.

Note The storage area name must start with an alpha character (A . . Z).

VALUES: Any valid 31-character Rdb storage area name

DEFAULT: SMALL_TABLE_AREA

(35) SMALL_SORTED

The **SMALL_SORTED** parameter is used to set the common storage area name for sorted indexes of tables that have a record count less than the SA_MIN_CARD parameter. **IT IS HIGHLY RECOMMENDED THAT YOU PRECEDE THE "SMALL_SORTED" NAME WITH THE NAME OF THE DATABASE IF STORING MORE THAN ONE DATABASE IN THE SAME DIRECTORY!** For example, if storing the INVOICE database and the ORDERS database in the same directory, the values for this parameter might be "INVOICE_SMALL_SORTED" or "ORDERS_SMALL_SORTED," depending on which database was being tuned during a session.

Note The storage area name must start with an alpha character (A . . Z).

VALUES: Any valid 31-character Rdb storage area name

DEFAULT: SMALL_SORTED_AREA

(36) SMALL_HASHED

The **SMALL_HASHED** parameter is used to set the common storage area name for hashed indexes of tables that have a record count less than the SA_MIN_CARD parameter**. IT IS HIGHLY RECOMMENDED THAT YOU PRECEDE THE "SMALL_HASHED" NAME WITH THE NAME OF THE DATABASE IF STORING MORE THAN ONE DATABASE IN THE SAME DIRECTORY!** For example, if storing the INVOICE database and the ORDERS database in the same directory, the values for this parameter might be "INVOICE_SMALL_HASHED" or "ORDERS_SMALL_HASHED," depending on which database was being tuned during a session.

Note The storage area name must start with an alpha character (A . . Z).

VALUES: Any valid 31 character Rdb storage area name

DEFAULT: SMALL_HASHED_AREA

(37) TUNE_FOR_COMPRESSION

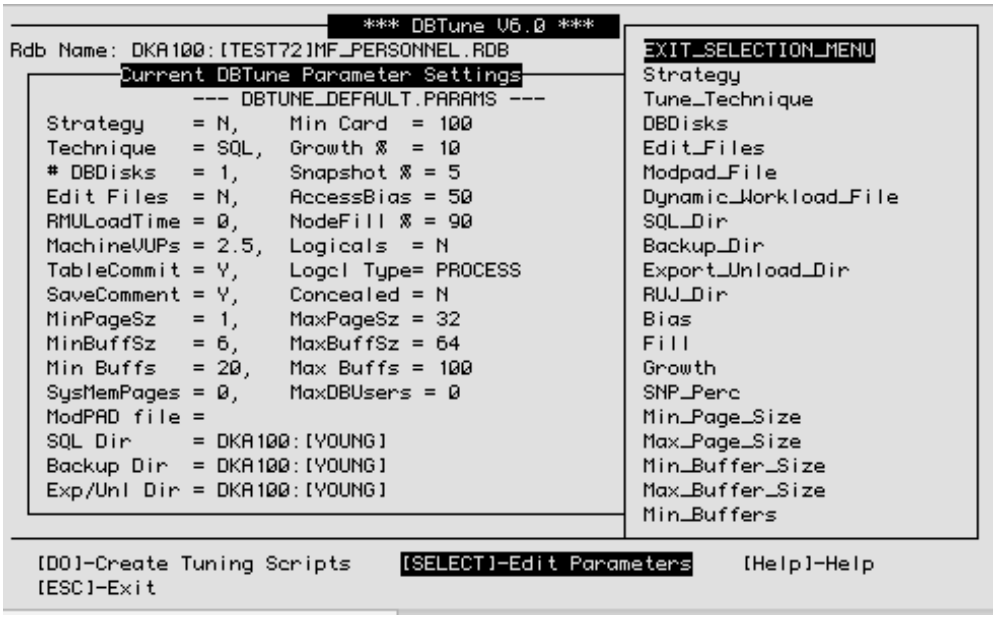
The **TUNE_FOR_COMPRESSION** parameter controls the values used for tuning storage areas: compressed or uncompressed data values. The default is to use compressed (the current Rdb 7 default) data values. If you want uncompressed values to be used for tuning, this parameter can be changed to **N**. Tuning using compressed values may save disk space and will use the average compressed record sizes for those tables that have compression enabled. However, tuning based on compressed values may also result in increased record fragmentation as the database is used. In addition, using compressed values requires more processing time by DBTune than using uncompressed values (see the note below). Tuning using uncompressed values will reduce record fragmentation during database usage and is quicker to calculate.

Note Setting the value of this parameter to Y will require more processing time for DBTune to determine actual compression values of data stored in the database. For databases that are several gigabytes in size, this option could add several hours of processing time to DBTune.

VALUES: **Y** **YES**, Tune using COMPRESSED data values
N **NO**, Tune using UNCOMPRESSED data values

DEFAULT: **Y**

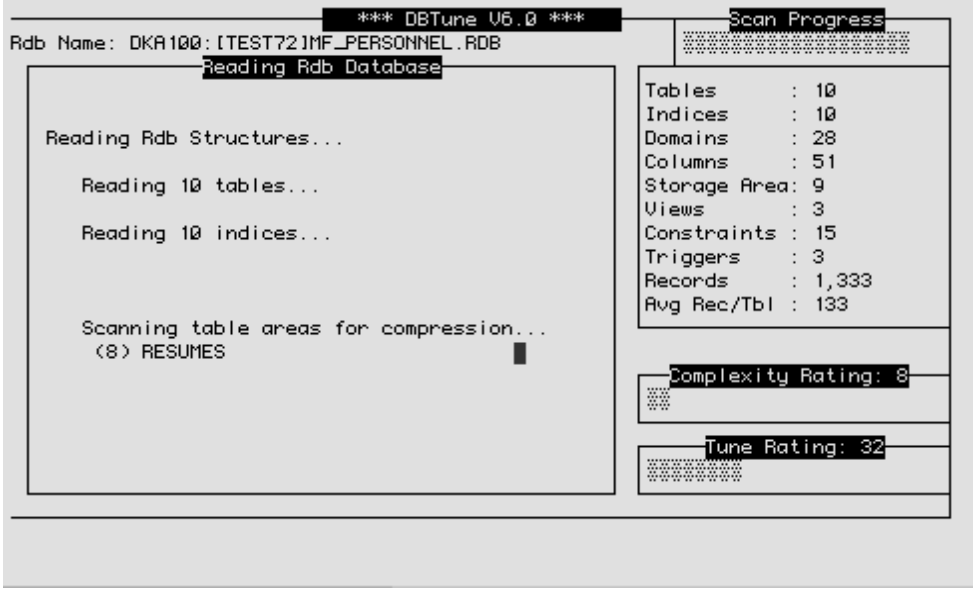
After the DBTune parameter file has been read in and validated, the values are displayed for acceptance or online editing:



Note The actual parameter values used for a DBTune session are recorded in the DBTune **LOG** file.

Step 3: Read Database Structure

DBTune automatically loads the database structure. There is no input required for this phase. The status window provides progress information during this process.



Step 4: Read Customized Analysis (MODPAD) / Workload Data

If a valid filename was specified for the DBTune parameter **MODPAD_FILE** or the parameter **DYNAMIC_WORKLOAD_FILE**, those files are parsed for validity. In the case of the **MODPAD_FILE**, any variable data (columns containing an "@" character) is filled in with the actual values for the database. (For more information on variable data, see explanation of the **MODPAD_FILE** parameter on page 138.) The results of this step are used to create DBTune's Performance Analysis Data (PAD) file. Any errors that occur during this step will be placed in the DBTune log file.

Note If DBTune detects errors when parsing the Dynamic Workload or ModPAD file you will be given the opportunity to view a log of the errors found and to make corrections. You may also elect to continue without making corrections, allowing DBTune to substitute default values for erroneous settings.

Step 5: Generate Performance Analysis Data (PAD) File

DBTune creates the **Performance Analysis Data** (PAD) file using the logical and physical database design, dynamic activity, transaction analysis results, DBTune parameters, and ModPAD file. You can dynamically change all column values in the PAD file during an editing session except for those columns below that explicitly state "DO NOT CHANGE!" The PAD file consists of five sections:

DBDISKS

Assignment of physical disk devices to be used when spreading storage areas. It contains three values:

Logical Disk Name DO NOT CHANGE!

Physical Specification Disk and directory specification for the logical disk.

Available Blocks Disk blocks available for database storage areas.

TABLE

Information for database tables used during tuning. It contains seven values:

Table Name DO NOT CHANGE!

Cardinality Current number of rows in table. Can be modified to desired volume.

Growth Cardinality will increase (grow) by this percentage.

Access Bias 0..100, 0 = 100% Write bias, 100 = 100% Read bias. 50 = 50% Write, 50% Read.

Activity Level 1-9, 1 indicates the lowest activity, 9 indicates the highest activity. Used to optimize disk utilization (if STOR_AREA_SPREAD is A or B).

Snapshot % Allocation for table's snapshot file will be set to a percentage of the table's data (.**RDA**) file allocation.

Tune Table Y/N, Tune the table? Used only when performing RMU/LOADs (**TUNE_TECHNIQUE** parameter = **RMU**). If performing **RMU/LOAD**s and Tune Table is set to **N**, table will be skipped during the tuning process.

Enable Compression Y/N, Enable Compression? Will enable (Y)/disable (N) compression for the individual table. By default, this is set to the current compression setting for the table in the database.

INDEX

Information for database indexes used during tuning. It contains eight values:

Index Name DO NOT CHANGE!

Index Type S/H/R/O (**S**orted, **H**ash Scattered,Sorted **R**anked, Hash **O**rdered). This value may be changed to alter the index type.

Activity Level 1-9, 1 indicates the lowest activity, 9 indicates the highest activity. Used to optimize disk utilization (if **STOR_AREA_SPREAD** is A or B).

Average Duplicates Current number of average duplicates for the index.

Key Values/Node Indicates the number of records that will fit in the node size that DBTune will calculate for this sorted index. Can be changed to increase or decrease node size.

Node Fill Fill factor to be used for the sorted index.

Snapshot % Allocation for index's snapshot file will be set to a percentage of the index's data (.**RDA**) file allocation.

Tune Index Y/N, Tune the index? Used only when performing **RMU/LOAD**s (**TUNE_TECHNIQUE** parameter = **RMU**). If performing **RMU/LOAD**s and Tune Index is set to **N**, index will be skipped during the tuning process.

CLUSTER

Used to indicate tables and/or indexes to be clustered together in the same storage area. May also specify that a table is to be **PLACED VIA** an index.

To Cluster two objects together:
[EMPLOYEES] & {EMP_EMPLOYEE_ID}

To have a table PLACED VIA an index:
[EMPLOYEES] = {EMP_EMPLOYEE_ID}

Note It is not necessary to cluster a table and an index together to use the **PLACE VIA** option. When a table is **PLACED VIA** an index, the page size is calculated to hold the number of table records equal to the average number of duplicates for the specified index. For example, if the EMPLOYEE_HISTORY table is PLACED VIA the EMPLOYEE_HIST_IDX and the EMPLOYEE_HIST_IDX has an average duplicate value of 15, then the page for the EMPLOYEE_HISTORY table will be sized to hold at least 15 records.

CONTRA

Used to indicate tables and/or indexes that are NOT to be stored on the same disk.

To indicate that the storage areas for 2 objects should be on separate disks: [EMPLOYEES]~[RESUMES]. See description/examples below

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
! PAD (On-Line Performance Analysis Data--DBTune V6.0)
!
! Listed below are values that will be used for tuning storage areas
! for database:  MF_PERSONNEL
! The user may change values for those columns underlined with '--*--'.
! When finished, exit and save this file to continue the DBTune process.
!
! Following are RESERVED characters and their interpretations:
!  '!'     : comment line (if ! is the 1st char, entire line ignored)
!  '/'     : column separator, MUST exist between column values,
!            e.g. col1/col2/.../
!  '[]'    : the enclosed item is a database TABLE, e.g., [EMPLOYEE_TABLE]
!  '{}'    : the enclosed item is a database INDEX, e.g., {EMPLOYEE_INDEX}
!  '&'     : CLUSTER symbol, the item following this symbol will be stored
!            in the same storage area as the item preceding this symbol
!  '='     : PLACE VIA symbol, [table_name]={index_name} indicates to PLACE
!            the table VIA the index (index must belong to the table)
!  '~'     : CONTRA symbol, the item following this symbol will NOT be
!            stored on the same disk as the item preceding this symbol
!  '*DISK' : indicates this is a physical disk assignment line
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
! DBDISKS Section:
! ================
```

You may edit the **PAD** file online during the DBTune process to customize the performance analysis criteria for their particular applications and their environment. This can be accomplished by setting the **EDIT_FILES** parameter to **Y**. In addition to the typical transaction parameters such as workload and volume data, the performance analysis considers the logical design to create an optimized Rdb physical design. Thus, the database is physically structured to achieve optimal performance according to its planned and actual usage.

Note If DBTune detects errors when parsing the **PAD** file, you will be given the opportunity to view a log of the errors found and to make corrections. You may also elect to continue without making corrections, allowing DBTune to substitute default values for erroneous settings.

The **PAD** file may be generated anew every time DBTune is run or the **PAD** file can be "saved" in a **ModPAD** file in order to reproduce a given tuning strategy. To control how much of a **PAD** is generated each time DBTune is run, the following parameters can be set:

MODPAD_FILE

If left blank, DBTune generates the **PAD** file using default or parameter values. If assigned a valid file specification, DBTune uses the named ModPAD file to "seed" the values in the new **PAD** file, substituting actual database values where variable data is indicated ("@") and using hard-coded values otherwise. If assigned an invalid file specification (ie the file does not currently exist), DBTune will generate a ModPAD file with the given name, and sunbstitute default (ie "@") values for all appropriate entries. You may them edit this file for subsequent runs.

DYNAMIC_WORKLOAD_FILE

If a valid file specification is assigned to this parameter, activity information in the named file will be used as input to the **PAD** file and will override any activity information found in the **MODPAD_FILE**.

The following characters are considered **RESERVED**, and DBTune uses them to interpret the **Performance Analysis Data** file:

RESERVED	INTERPRETATION
!	Comment line, entire line ignored
/	Column separator, e.g., col1/col2/…/
***DISK**	Indicates this is a physical disk assignment line, e.g., *DISK01/DISK1:[MYDATA.RDB]/2500/
[ ]	Enclosed item is an Rdb table name (except in the case of a disk assignment), e.g., [EMP_TABLE]
{ }	Enclosed item is an Rdb index name, e.g., {EMP_INDEX}
&	Used to **CLUSTER** two items in the same storage space e.g., [TABLE_1] & [INDEX_2]
=	Used to specify that a table is to be **PLACED VIA** an index, e.g., [INVOICE_TABLE]=[INDEX_IDX]
~	Used to force two items to be stored on separate disks (a "**CONTRA**"), e.g., [TABLE_1]~-[TABLE_2]

The following is a sample listing of a typical PAD file:

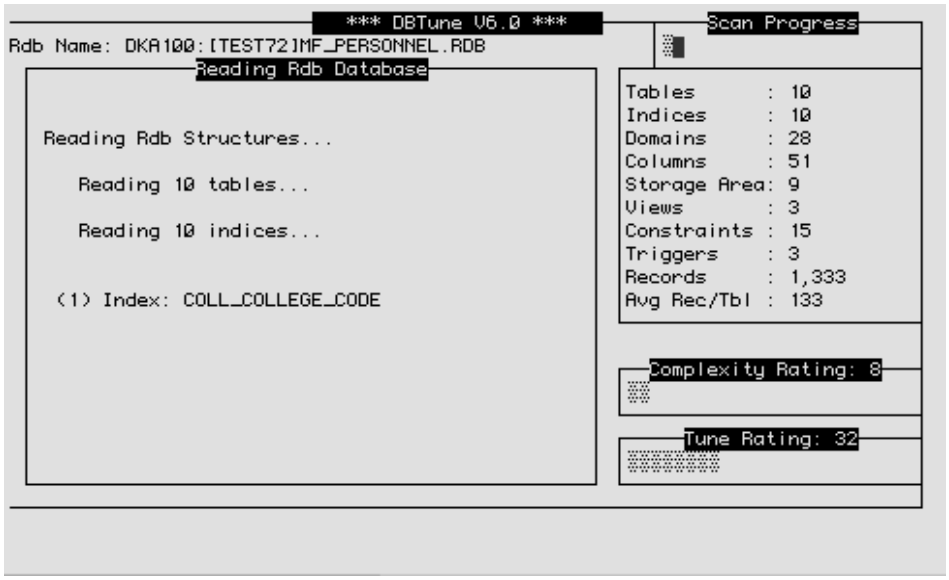
```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
! PAD (On-Line Performance Analysis Data--DBTune V6.0)
!
! Listed below are values that will be used for tuning storage areas
! for database:  MF_PERSONNEL
! The user may change values for those columns underlined with '--*--'.
! When finished, exit and save this file to continue the DBTune process.
!
! Following are RESERVED characters and their interpretations:
!  '!'     : comment line (if ! is the 1st char, entire line ignored)
!  '/'     : column separator, MUST exist between column values,
!            e.g. col1/col2/.../
!  '[]'    : the enclosed item is a database TABLE, e.g., [EMPLOYEE_TABLE]
!  '{}'    : the enclosed item is a database INDEX, e.g., {EMPLOYEE_INDEX}
!  '&'     : CLUSTER symbol, the item following this symbol will be stored
!            in the same storage area as the item preceding this symbol
!  '='     : PLACE VIA symbol, [table_name]={index_name} indicates to PLACE
!            the table VIA the index (index must belong to the table)
!  '~'     : CONTRA symbol, the item following this symbol will NOT be
!            stored on the same disk as the item preceding this symbol
!  '*DISK' : indicates this is a physical disk assignment line
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
! DBDISKS Section:
! ================
!
! The parameter 'DBDISKS' indicates 1 disk is available for use.
! For each disk, a physical DISK:[DIR] specification and available free
! blocks can be entered.  If a disk and directory are NOT entered for each
! disk, the disk and directory where the .RDB file resides will be used
! (i.e., DKA100:[TEST72]).
! The number of free blocks for each disk is set to 'unlimited' by default.
! If the physical disk specification you enter is accessible to this
! process and is set to 'unlimited', the free space will be derived
! automatically.  Or, you can set your own limit to the free space that
! will be used by entering a number in the 'Free Blocks' column below.
!
! Following are two examples:
! DISK001 / $1$DUA1:[DATA.RDB] / unlimited /  <-- will be calc'd during tuning
! DISK002 / $1$DUA2:[MORE_DATA]/    150000 /  <-- will limit to 150,000 blocks
!
! * * * * * * * * * * * * * * * *  NOTE  * * * * * * * * * * * * * * * *
! Enter physical disk and directory specifications in order of disk speed
! and disk utilization:  the fastest disk that has the least amount
! of I/O requests should be entered for 'DISK001', the next fastest disk
! should be 'DISK002', etc.
!
!                       Physical Disk & Directory              Free Blocks
!           ------------------------*------------------------ ------*------
*DISK001   /DKA100:[TEST72]                                   /0000068857840/
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
! TABLE Section:
! ==============
!
! 'Num Recs' column: Table Cardinality; used to calculate allocation
! 'Grw %' column   : Table Growth Percentage (values: 0..999)
! 'Acc Bia' column : Access Bias (values 0..100; 0=100% Write, 100=100% Read)
```

```
! 'Act Lvl' column : Table Activity Level (values: 1..9 with 1=Low,9=High)
! 'Snp %' column   : Snapshot percentage (values: 0..500)
! 'Tun Tbl' column : Tune Table? (values: Y-Yes, N-No; only for RMU/LOADs)
! 'Ena Cmp' column : Enable Compression? (values: Y-Yes, N-No)
!
! Any of the following table information (except the TABLE NAME) can be
! changed to customize the Performance Analysis and tuning processes.
!                                  Number    Grw Acc Act Snp Tun Ena
! Database Table Name              of Recs    %  Bia Lvl  %  Tbl Cmp
! -------------------------------- -----*------ -*- -*- -*- -*- -*- -*-
[ CANDIDATES ]                    /           3/ 10/50 / 5 /  5/ Y / Y /
[ COLLEGES ]                      /          15/ 10/50 / 5 /  5/ Y / Y /
[ DEGREES ]                       /         165/ 10/50 / 5 /  5/ Y / Y /
[ DEPARTMENTS ]                   /          26/ 10/50 / 5 /  5/ Y / Y /
[ EMPLOYEES ]                     /         100/ 10/50 / 5 /  5/ Y / Y /
[ JOBS ]                          /          15/ 10/50 / 5 /  5/ Y / Y /
[ JOB_HISTORY ]                   /         274/ 10/50 / 5 /  5/ Y / Y /
[ RESUMES ]                       /           3/ 10/50 / 5 /  5/ Y / Y /
[ SALARY_HISTORY ]                /         729/ 10/50 / 5 /  5/ Y / Y /
[ WORK_STATUS ]                   /           3/ 10/50 / 5 /  5/ Y / Y /
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
! INDEX Section:
! ==============
!
! 'Idx Typ' column : Index Type (values: S-Sorted, R-Ranked,
!                                    H-Scattered, O-Ordered)
! 'Act Lvl' column : Index Activity Level (values: 1..9 with 1=Low, 9=High)
! 'Avg Dups' column: Average Duplicates for the index (values: 0..99999)
! 'Key Nod' column : Index Key Values Per Node (values: 3..999)
! 'Fil %' column   : Index Node Fill Percentage (values: 33..100)
! 'Snp %' column   : Snapshot Percentage (values: 0..500)
! 'Tun Idx' column : Tune Index? (values: Y-Yes, N-No; only for RMU/LOADs)
!
! Any of the following index information (except the INDEX NAME) can be
! changed to customize the Performance Analysis and tuning processes.
!                                  Idx Act  Avg  Key Fil Snp Tun
! Database Index Name              Typ Lvl  Dups Nod  %   %  Idx
! -------------------------------- -*- -*- --*-- -*- -*- -*- -*-
{ COLL_COLLEGE_CODE }             / S / 5 /    0/ 33/ 90/  5/ Y /
{ DEG_COLLEGE_CODE }              / S / 5 /   12/ 33/ 90/  5/ Y /
{ DEG_EMP_ID }                    / S / 5 /    1/ 33/ 90/  5/ Y /
{ DEPARTMENTS_INDEX }             / S / 5 /    0/ 33/ 90/  5/ Y /
{ EMPLOYEES_HASH }                / H / 5 /    0/ - / - /  5/ Y /
{ EMP_EMPLOYEE_ID }               / S / 5 /    0/ 33/ 90/  5/ Y /
{ EMP_LAST_NAME }                 / S / 5 /    1/ 33/ 90/  5/ Y /
{ JH_EMPLOYEE_ID }                / S / 5 /    2/ 33/ 90/  5/ Y /
{ JOB_HISTORY_HASH }              / H / 5 /    2/ - / - /  5/ Y /
{ SH_EMPLOYEE_ID }                / S / 5 /    7/ 33/ 90/  5/ Y /
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
! CLUSTER Section:
! ================
!
! Enter below specific pairs of items that you wish to be clustered together
! in the same storage area.  The first item in the pair will determine the
! storage area to be used.  So, to cluster three tables together in the
! same storage area, the following two lines could be entered:
!   [TABLE_A] & [TABLE_B] meaning 'store TABLE_B in TABLE_A's storage area'
!   [TABLE_A] & [TABLE_C] meaning 'store TABLE_C in TABLE_A's storage area'
! Only two items can be listed per line and they must be separated by an '&'.
```

```
! Tables that are to be PLACED VIA an index can also be entered in this
! section.  The table must be listed first, followed by an '=' and then
! an index.  The index must belong to the table and the PLACE VIA line
! must follow the format: [table_name]={index_name}.
! REMEMBER: Tables are surrounded by []'s and indexes are surrounded by {}'s.
!
! Enter pairs of items to be CLUSTERED and tables to be PLACED VIA an index:
! ------------------------------------------------------------------------
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
! CONTRA Section:
=================
!
! Enter below specific pairs of items that you do NOT wish to be stored on
! the same disk (a pair of such items is called a 'CONTRA').  For example,
! to ensure that a particular table and index are not stored on the same
! disk, the following line could be entered (without the '!'):
!   [TABLE_A] ~ {INDEX_B}
! meaning 'do not store TABLE_A and INDEX_B on the same disk.'
! RDA's and SNP's are automatically stored on separate disks, if possible,
! so the user does not have to specify a CONTRA for this type of condition.
! Only two items can be listed per line and they must be separated by a '~'.
! REMEMBER: Tables are surrounded by []'s and indexes are surrounded by {}'s.
!
! Do NOT store the following pairs of items on the same disk:
! ------------------------------------------------------------------------
!
```

Step 6: Performance Analysis

The **Performance Analysis** reads the **PAD** file and the logical and physical design of Rdb. Then it combines these inputs to create a new physical design for the Rdb database. This design is optimized to increase performance by considering the interrelations of the database and its application usage and operating environment. The applications can be used as they are without application programming changes. Recommendations for setting affected **VMS AUTHORIZE** or **SYSGEN** quotas are provided in the **REVIEW_AND_GUIDE.REPORT** created by DBTune for this design. There is no input required for this phase. The status window provides progress information on the tuning process.



Step 7: Generate Disk Utilization (DISKUTIL) File

The **Disk Utilization** file is created after the performance analysis and database tuning have optimized the new design. All “new” or “relocated” storage areas are assigned to one of the logical (DBDISK) devices while “existing” areas are placed in their original locations and will not show up in this file. In addition to listing which storage areas are assigned to which disk, the total blocks assigned to each device is given.

If the **EDIT_FILES** parameter is set to **Y**, DBTune pauses and allows you to edit the device assignments. If **EDIT_FILES = N**, the editing session will be skipped.

Note If DBTune detects errors when parsing the Disk Utilization file, you will have the opportunity to view a log of the errors found and to make corrections. You may also elect to continue without making corrections, allowing DBTune to substitute default values for erroneous settings.

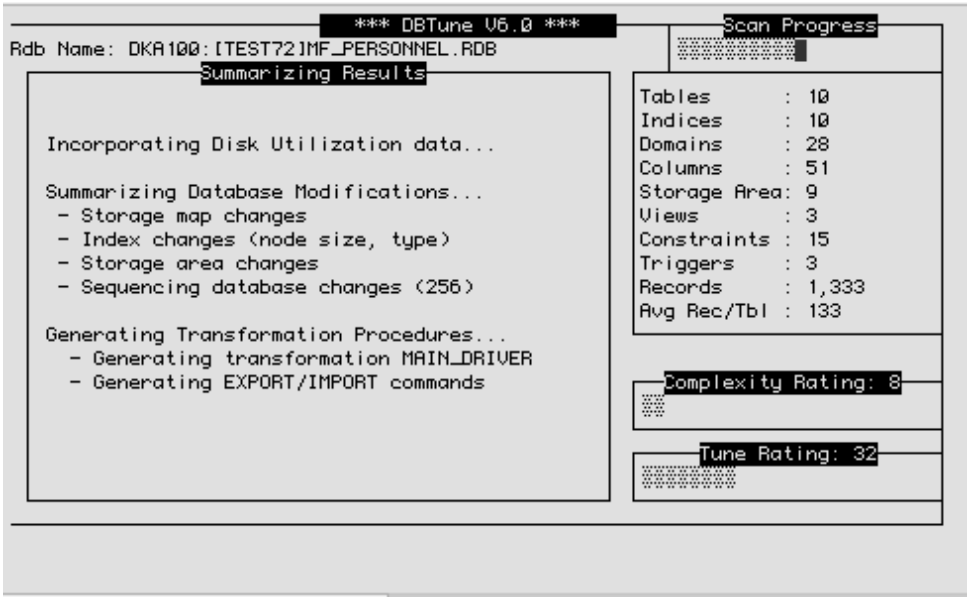
Note After tuning calculations are complete, the disk utilization file is edited. You must ensure that there will be sufficient space to hold any storage areas that have been reassigned. DBTune will not override any assignments that are made during this edit session.

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!
! Disk Utilization Data  (DBTune V6.0)
!
! Listed below are values used to control the placement of 'new' and
! 'relocated' storage areas for the database: 'MF_PERSONNEL'.
! The Physical Disk & Directory specification and Free Blocks are listed
! again for the DBDISKS, but the values in these columns cannot be changed.
! The only values that can be changed in this file are those in the 'Disk'
! column, which can be found in the STORAGE AREA ASSIGNMENT section below.
! After editing, exit and save this file to continue the DBTune process.
!
! Following are the disk specifications chosen previously:
!
!                       Physical Disk & Directory               Free Blocks
!           ------------------------------------------------- -------------
!DISK001   /DKA100:[TEST72]                                  / 68822900/
!
! STORAGE AREA ASSIGNMENT Section:
! ================================
!
! Following are disk assignments for storage areas that are new or are being
! relocated.  The 'Disk'column may be changed by the user, but the user is
! responsible for ensuring adequate disk space exists for any changes made.
! Each line that begins with a '!' is considered a comment and will be ignored.
! Every other line will be considered a storage area disk assignment.
```

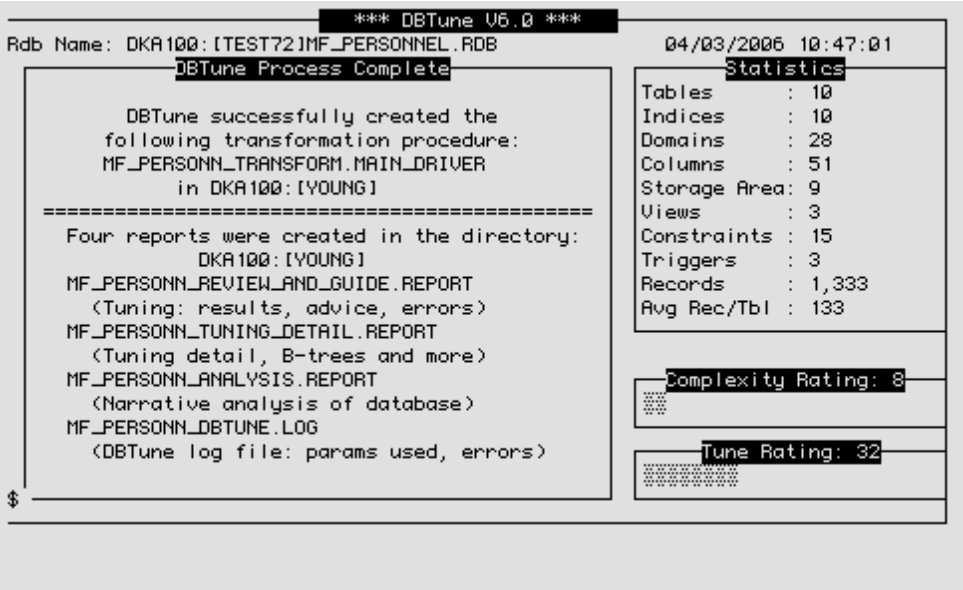
```
!
!                                File  Allocation
! Storage Area File              Type   (blocks)        Disk
!------------------------------- ---- ------------ -----*-----
 MF_PERSONNEL                   /RDB /          290/ DISK001   /
 DEG_COLLEGE_CODE_IDX           /RDA /           64/ DISK001   /
 DEG_EMP_ID_IDX                 /RDA /           52/ DISK001   /
 EMPIDS_LOW                     /RDA /          117/ DISK001   /
 EMPIDS_MID                     /RDA /          129/ DISK001   /
 EMPIDS_OVER                    /RDA /           96/ DISK001   /
 EMP_EMPLOYEE_ID_IDX            /RDA /           48/ DISK001   /
 EMP_INFO                       /RDA /           42/ DISK001   /
 EMP_LAST_NAME_IDX              /RDA /           72/ DISK001   /
 JH_EMPLOYEE_ID_IDX             /RDA /          104/ DISK001   /
 LIST_AREA                      /RDA /          528/ DISK001   /
 RDB$SYSTEM                     /RDA /         3066/ DISK001   /
 SALARY_HISTORY                 /RDA /           78/ DISK001   /
 SH_EMPLOYEE_ID_IDX             /RDA /          112/ DISK001   /
 SMALL_SORTED_AREA              /RDA /           57/ DISK001   /
 SMALL_TABLE_AREA               /RDA /          144/ DISK001   /
 DEG_COLLEGE_CODE_IDX           /SNP /           24/ DISK001   /
 DEG_EMP_ID_IDX                 /SNP /           24/ DISK001   /
 EMPIDS_LOW                     /SNP /           18/ DISK001   /
 EMPIDS_MID                     /SNP /           18/ DISK001   /
 EMPIDS_OVER                    /SNP /           18/ DISK001   /
 EMP_EMPLOYEE_ID_IDX            /SNP /           24/ DISK001   /
 EMP_INFO                       /SNP /           18/ DISK001   /
 EMP_LAST_NAME_IDX              /SNP /           36/ DISK001   /
 JH_EMPLOYEE_ID_IDX             /SNP /           24/ DISK001   /
 LIST_AREA                      /SNP /           72/ DISK001   /
 RDB$SYSTEM                     /SNP /          154/ DISK001   /
 SALARY_HISTORY                 /SNP /           18/ DISK001   /
 SH_EMPLOYEE_ID_IDX             /SNP /           24/ DISK001   /
 SMALL_SORTED_AREA              /SNP /           18/ DISK001   /
 SMALL_TABLE_AREA               /SNP /           18/ DISK001   /
!                                    ==============
! * DISK001   :    30 Stor File(s),  Used: 5507 out of 68822900 blocks
!
!                                    ==============
!** Placed:        30 Stor File(s),  Used: 5507 total blocks
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

Step 8: Generate Rdb Transformation Procedure

DBTune automatically generates a procedure that you can then execute at a later time to transform the original database into its tuned structure. There is no input required for this phase. The status window provides progress information on the generation process.



Upon completion, the transformation procedure, advice, tuning, and documentation files are presented.



Step 9: Transform Rdb

The output created by the DBTune process consists of a set of reports and a transformation procedure that tunes the database. **IT IS STRONGLY SUGGESTED THAT THE REPORTS BE REVIEWED PRIOR TO EXECUTION OF THE TRANSFORMATION PROCEDURE**. These reports provide an analysis of the database as well as tuning advice, warnings about potential problems, and instructions on how to execute the transformation procedure. Examples of these reports can be found on the following pages.

When all the reports have been reviewed, you must ensure that any VMS logicals currently required by the database have been assigned prior to the database transformation or the transformation procedure may fail!

After reviewing the reports and assigning any necessary database logicals, the database can be tuned by executing the transformation **MAIN_DRIVER** command file. This command file is called **<<database_id>>_TRANSFORM.MAIN_DRIVER** and can be found in the directory assigned to the **SQL_DIR** parameter. The **MAIN_DRIVER** is a **DCL** command file that will assign necessary VMS logicals and execute DCL and SQL scripts to tune the target database.

The **MAIN_DRIVER** command file can be executed either online or in batch. If executed online, the procedure asks a series of questions and provides warnings if privilege is insufficient or if special actions need to be taken prior to tuning. **ALL USERS MUST BE OUT OF THE DATABASE WHEN THE MAIN_DRIVER IS EXECUTED OR THE SCRIPTS WILL FAIL!**

Following is an example of an execution of the **MAIN_DRIVER** procedure for the MF_PERSONNEL database:

```
$ set nover
$!*****************************************************************************
$!
$! FILE   : MF_PERSONN_TRANSFORM.MAIN_DRIVER
$! CREATED: 03/27/2006 15:49:24  (DBTune V6.0)
$!
$! Main driver to alter the database for...
$!
$!        DATABASE NAME: MF_PERSONNEL
$!
$! This is a DCL command file that is to be executed at the '$' prompt by
$! typing      @MF_PERSONN_TRANSFORM.MAIN_DRIVER
$! This driver command file will execute SQL command files to
$! modify your database.
$!
$!*****************************************************************************
$ set on
$ on error then goto CLEANUP
$ on severe_error then goto CLEANUP
$ on warning then goto CLEANUP
$ say :== write sys$output
$!
$! If Batch mode, skip user prompts ...
$ if "''f$mode()'" .eqs. "BATCH" then -
     goto CHECK_EXPUNL
$!
$INTRO:
$ type sys$input

================================================================================

  You are executing the MF_PERSONN_TRANSFORM.MAIN_DRIVER command file.
  This command file will execute scripts to modify the following database:
                   MF_PERSONNEL

  If this database was previously defined using VMS logicals, those
  logicals MUST exist for this process or the scripts will fail.

================================================================================

$ read/end=EXIT/error=EXIT -
  /prompt="Press <RETURN> to continue, <CTRL>-Z to exit..." sys$command YESNO
$!
$CHECK_EXPUNL:
$ DISK_NM = f$parse("DKA100:[YOUNG]",,,"DEVICE")
$ if "''DISK_NM'" .nes. "" .and. f$getdvi(DISK_NM,"DEVCLASS") .eq. 1
$   then
$     if f$getdvi(DISK_NM,"FREEBLOCKS") .ge. 1255 then goto CHECK_SQLDIR
$ type sys$input
```

```
================================================================================

  Insufficient FREE BLOCKS are available on the disk on which the
  SQL EXPORT file will be placed.  The following free space is
  required before tuning can continue:

$ SAY "    Disk: ''DISK_NM'  Free Blocks Required: 1255"
$ SAY ""
$ SAY "  The DBTUNE Main Driver will now exit..."
$ SAY "========================================================================"
$ SAY ""
$ goto EXIT
$   endif
$!
$CHECK_SQLDIR:
$ DISK_NM = f$parse("DKA100:[YOUNG]",,,"DEVICE")
$ if "''DISK_NM'" .nes. "" .and. f$getdvi(DISK_NM,"DEVCLASS") .eq. 1
$   then
$     if f$getdvi(DISK_NM,"FREEBLOCKS") .ge. 1000 then goto NOTIFY_BACKUP
$ type sys$input

================================================================================

  Insufficient FREE BLOCKS are available on the tuning workspace
  disk.  At least 1000 blocks of free space are needed for scratch
  files on the following disk before tuning can continue:

$ SAY "            Disk: ''DISK_NM'"
$ SAY ""
$ SAY "  The DBTUNE Main Driver will now exit..."
$ SAY "========================================================================"
$ SAY ""
$ goto EXIT
$   endif
$!
$NOTIFY_BACKUP:
$ type sys$input

================================================================================

  For this tuning session, an SQL EXPORT/IMPORT will be performed
  on your database.  Before doing so, however, an RMU/BACKUP of
  your database will be taken and placed in the backup directory:
         DKA100:[YOUNG]
$ DISK_NM = f$parse("DKA100:[YOUNG]",,,"DEVICE")
$ if "''DISK_NM'" .nes. "" .and. f$getdvi(DISK_NM,"DEVCLASS") .eq. 1
$    then
$      if f$getdvi(DISK_NM,"FREEBLOCKS") .ge. 2652 then goto BCK_SPACE_OK
$ type sys$input

  Insufficient FREE BLOCKS are available on the disk on which the
  RMU/BACKUP file will be placed.  The following free space is
  required before tuning can continue:

$ SAY "    Disk: ''DISK_NM'  Free Blocks Required: 2652"
$ SAY ""
$ SAY "  The DBTUNE Main Driver will now exit..."
$ SAY "========================================================================"
$ SAY ""
$ goto EXIT
$ endif
$!
```

```
$BCK_SPACE_OK:
$ type sys$input

================================================================================

$ if "''f$mode()'" .eqs. "BATCH" then -
     goto DRIVER_SETUP
$ read/end=EXIT/error=EXIT -
  /prompt="Press <RETURN> to continue, <CTRL>-Z to exit..." sys$command YESNO
$!
$WARN_QUIET:
$ type sys$input

================================================================================

  WARNING:
  --------
  If this database was manually opened with an RMU/OPEN command, it
  must be closed on all nodes with an RMU/CLOSE/CLUSTER command
  before continuing, otherwise database modifications may fail.
  The modifications may also fail if there are users or BATCH
  processes that are accessing the database during execution
  of these scripts.  If either of these situations exist, exit
  this MAIN_DRIVER procedure now and correct before continuing.

================================================================================

$ read/end=CLOSE_MSG/error=CLOSE_MSG -
  /prompt="Press <RETURN> to continue execution, <CTRL>-Z to exit..." sys$command YESNO
$ goto DRIVER_SETUP
$!
$CLOSE_MSG:
$ type sys$input

  To see if there are any users/processes which are currently
  accessing this database, issue the following command:
     RMU/DUMP/USERS DKA100:[TEST72]MF_PERSONNEL

  To ensure the database is closed for tuning, issue the command:
     RMU/CLOSE/CLUSTER DKA100:[TEST72]MF_PERSONNEL

$ exit
$!
$!
$DRIVER_SETUP:
$ type sys$input

          ================================================

              No more input is required from the user.

                Execution of scripts beginning...

          ================================================

$ ALI_START = f$time()
$ if f$trnlnm("ALI_RDB_DATABASE", "LNM$PROCESS") .nes. "" then -
     deassign/process ALI_RDB_DATABASE
$ assign/nolog/job DKA100:[TEST72]MF_PERSONNEL.RDB ALI_RDB_DATABASE
$ assign/nolog/job ITA$0:[DBTRDBITA60] ALI_SQL_HOOK
$! Make sure the SQL error checking routine exists in the ALI_SQL_HOOK directory...
$ FILECHECK = f$search("ALI_SQL_HOOK:CHECK_FOR_SQL_ERRORS.COM")
$ if FILECHECK .eqs. "" then goto BADCHECK
```

```
$! Set up the SQL symbol...
$ SQL :== "$SQL$"
$! Set up the SQL error-checking logical...
$ assign/nolog/job "Y" ALI_SQL_ERROR
$ if "''DELETE'" .nes. ""
$   then
$   HOLD_DELETE = DELETE
$   DELETE :== "DELETE"
$   endif
$!
$! Set some Rdb logicals.
$ define/nolog/job RDM$BIND_BUFFERS 1000
$ define/nolog/job RDMS$RUJ DKA100:[YOUNG]
$ define/nolog/job RDM$BIND_RUJ_EXTEND_BLKCNT 9999
$!
$! Set the RMS extend quantity
$ SET RMS_DEFAULT/EXTEND_QUANTITY=50000
$!
$ DEFAULT_DIR = f$environment("DEFAULT")
$ set def DKA100:[TEST72]
$!
$START:
$!
$RESTART:
$!
$CLOSE_DB:
$ set noon
$ type sys$input

  The database will now be closed with an RMU/CLOSE command.
  Any processes currently attached to the database will be exited.

$ spawn/wait/nolog/out=dbtune_close.out RMU/CLOSE/CLUSTER DKA100:[TEST72]MF_PERSONNEL
$ if f$search("dbtune_close.out") .nes. "" then -
    delete/nolog dbtune_close.out;
$!
$BACKUP:
$ set on
$ on error then goto BAD_BCK
$ on severe_error then goto BAD_BCK
$ on warning then goto BAD_BCK
$ type sys$input

  Performing an RMU/BACKUP of your database...

$ set ver
$ rmu/backup MF_PERSONNEL -
    DKA100:[YOUNG]MF_PERSONNEL.RBF
$ set nover
$ set noon
$!
$SAVE_PROT:
$ SAY ""
$ SAY "Creating a temporary work file which does not yet exist..."
$ SAY ""
$ if "''EDIT'" .nes. ""
$   then
$   HOLD_EDIT = EDIT
$   EDIT :== "EDIT"
$ endif
$!
$! Save ACL protection for the database...
$ EDIT/EDT/NOCOM DKA100:[YOUNG]RESTORE_ACL.TMP
```

```
EXIT
$ SET ACL/LIKE=(OBJECT_NAME=ALI_RDB_DATABASE) -
      DKA100:[YOUNG]RESTORE_ACL.TMP
if "''HOLD_EDIT'" .nes. "" then -
      EDIT :== "''HOLD_EDIT'"
$!
$ALTERS1:
$ type sys$input

 Before exporting, prepare the database for the import by making
 some preliminary changes...

 Executing the first set of ALTERS...
$ ASSIGN/JOB/NOLOG "Y" ALI_SQL_ERROR
$ SQL @DKA100:[YOUNG]MF_PERSONN.ALTERS1_SQL
$ if f$trnlnm("ALI_SQL_ERROR") .eqs. "Y" then goto FINISH
$!
$! Ensure that we have adequate working space on the disk
$! before continuing...
$!
$ DISK_NM = f$parse("DKA100:[YOUNG]",,,"DEVICE")
$ if "''DISK_NM'" .nes. "" .and. f$getdvi(DISK_NM,"DEVCLASS") .eq. 1
$   then
$     if f$getdvi(DISK_NM,"FREEBLOCKS") .ge. 1000 then goto EXPORT
$ say ""
$ say "============================================================================"
$ say ""
$ say "  Insufficient FREE BLOCKS are available on the tuning workspace"
$ say "  disk.  At least 1000 blocks of free space are needed for scratch"
$ say "  files on the following disk before tuning can continue:"
$ say ""
$ say "            Disk: ''DISK_NM'"
$ say ""
$ say "  The DBTUNE Main Driver will now exit..."
$ say "============================================================================"
$ say ""
$ goto EXIT
$   endif
$!
$EXPORT:
$ type sys$input

 Now, export the existing database and then delete it so that the newly
 altered database can be imported...

 EXPORTING the database...
$ ASSIGN/JOB/NOLOG "Y" ALI_SQL_ERROR
$ SQL @DKA100:[YOUNG]MF_PERSONN.EXPORT_SQL
$ @DKA100:[YOUNG]MF_PERSONN_APPEND.COM
$ if f$trnlnm("ALI_SQL_ERROR") .eqs. "Y" then goto FINISH
$!
$! Ensure that we have adequate working space on the disk
$! before continuing...
$!
$ DISK_NM = f$parse("DKA100:[YOUNG]",,,"DEVICE")
$ if "''DISK_NM'" .nes. "" .and. f$getdvi(DISK_NM,"DEVCLASS") .eq. 1
$   then
$     if f$getdvi(DISK_NM,"FREEBLOCKS") .ge. 1000 then goto DROP
$ say ""
$ say "============================================================================"
$ say ""
$ say "  Insufficient FREE BLOCKS are available on the tuning workspace"
$ say "  disk.  At least 1000 blocks of free space are needed for scratch"
```

```
$ say "  files on the following disk before tuning can continue:"
$ say ""
$ say "            Disk: ''DISK_NM'"
$ say ""
$ say "  The DBTUNE Main Driver will now exit..."
$ say "============================================================================="
$ say ""
$ goto EXIT
$   endif
$!
$DROP:
$ type sys$input

 DROPPING (deleting) the database...
$ ASSIGN/JOB/NOLOG "Y" ALI_SQL_ERROR
$ SQL @DKA100:[YOUNG]MF_PERSONN.DROP_SQL
$ @DKA100:[YOUNG]MF_PERSONN_APPEND.COM
$ if f$trnlnm("ALI_SQL_ERROR") .eqs. "N" then goto IMPORT
$ say ""
$ say "*******************************************************************************"
$ say ""
$ say "    ***** ERRORS OCCURRED DURING SQL EXECUTION! *****"
$ say ""
$ say "          Error occurred trying to DROP the database."
$ say ""
$ say "    *** Execution of SQL scripts did NOT complete successfully. ***"
$ say ""
$ say "*******************************************************************************"
$ goto CLEANUP
$IMPORT:
$!
$ type sys$input

 Now, import the database with the necessary changes...

 IMPORTING the database...
$ ASSIGN/JOB/NOLOG "Y" ALI_SQL_ERROR
$ SQL @DKA100:[YOUNG]MF_PERSONN.IMPORT_SQL
$ @DKA100:[YOUNG]MF_PERSONN_APPEND.COM
$ if f$trnlnm("ALI_SQL_ERROR") .eqs. "Y" then goto FINISH
$!
$! Ensure that we have adequate working space on the disk
$! before continuing...
$!
$ DISK_NM = f$parse("DKA100:[YOUNG]",,,"DEVICE")
$ if "''DISK_NM'" .nes. "" .and. f$getdvi(DISK_NM,"DEVCLASS") .eq. 1
$   then
$     if f$getdvi(DISK_NM,"FREEBLOCKS") .ge. 1000 then goto ALTERS2
$ say ""
$ say "============================================================================="
$ say ""
$ say "  Insufficient FREE BLOCKS are available on the tuning workspace"
$ say "  disk.  At least 1000 blocks of free space are needed for scratch"
$ say "  files on the following disk before tuning can continue:"
$ say ""
$ say "            Disk: ''DISK_NM'"
$ say ""
$ say "  The DBTUNE Main Driver will now exit..."
$ say "============================================================================="
$ say ""
$ goto EXIT
$   endif
$!
```

```
$ALTERS2:
$ type sys$input

 Perform any additional alters which need to be done after the import...

 Executing the second set of ALTERS...
$ assign/nolog/job "Y" ALI_SQL_ERROR
$ SQL @DKA100:[YOUNG]MF_PERSONN.ALTERS2_SQL
$ @DKA100:[YOUNG]MF_PERSONN_APPEND.COM
$!
$FINISH:
$ if f$trnlnm("ALI_SQL_ERROR") .nes. "Y" then goto FINISH_GOOD
$!
$ say ""
$ say "  ************************************************************************"
$ say ""
$ say "               ***** ERRORS OCCURRED DURING SQL EXECUTION! *****"
$ say ""
$ say "   Please review the MF_PERSONN.SQL_LOG file that is located in the directory"
$ say "                     DKA100:[YOUNG]"
$ say "   for the actual errors that occurred.  Be aware that some SQL script files"
$ say "   may have successfully executed before the error(s) were encountered."
$ say ""
$ say "      *** Execution of SQL scripts did NOT complete successfully. ***"
$ say ""
$ say "  ************************************************************************"
$ goto CLEANUP
$!
$FINISH_GOOD:
$ ALI_STOP = f$time()
$ say ""
$ say "  ******************************************************"
$ say ""
$ say "  *** Execution of SQL scripts completed successfully. ***"
$ say ""
$ say "     Tuning process began: ''ALI_START'"
$ say "     Tuning process ended: ''ALI_STOP'"
$ say ""
$ say "  ******************************************************"
$ type sys$input

 ========================================================================

  To restore ACL protection to the database, execute the following
  commands after successful completion of the MAIN_DRIVER scripts:

    $ SET DEFAULT DKA100:[TEST72]
  To restore ACL's for the Rdb root file:
    $ SET ACL/LIKE=(OBJECT_NAME=DKA100:[YOUNG]RESTORE_ACL.TMP) -
    _$ MF_PERSONNEL.RDB

 ========================================================================

$ goto CLEANUP
$!
$BADCHECK:
$! Print error message...
$ say "   ***"
$ say "   *** Error locating the file CHECK_FOR_SQL_ERRORS.COM in the ALI_DBTUNE_HOME"
$ CHECKLOG = f$trnlnm("ALI_SQL_HOOK")
$    if CHECKLOG .eqs. "" then goto BADLOG
$ say "   *** directory, cannot continue.  The logical ALI_DBTUNE_HOME is "
$ say "   *** assigned to ''CHECKLOG', which may be an invalid"
```

```
$ say "   *** directory or privilege may be insufficient."
$ say "   ***"
$ deassign/job ALI_SQL_HOOK
$ goto EXIT
$!
$BADLOG:
$ say "   *** directory, cannot continue.  The logical ALI_DBTUNE_HOME is currently"
$ say "   *** unassigned and needs to be assigned to the directory that contains"
$ say "   *** the DBTune command files."
$ say "   ***"
$ deassign/job ALI_SQL_HOOK
$ goto EXIT
$!
$BAD_BCK:
$ set nover
$ say " "
$ say "  An error occurred during the RMU/BACKUP.  No database changes"
$ say "  performed.  Exiting the MAIN_DRIVER procedure."
$ say " "
$!
$CLEANUP:
$! Clean up DBTune files generated during execution of the script...
$ deassign/job ALI_RDB_DATABASE
$ deassign/job ALI_SQL_HOOK
$ deassign/job ALI_SQL_ERROR
$ deassign/job RDMS$RUJ
$ deassign/job RDM$BIND_BUFFERS
$ deassign/job RDM$BIND_RUJ_EXTEND_BLKCNT
$ if f$search("DKA100:[YOUNG]COMMIT_OR_ROLLBACK.SQL") .nes. "" then -
     delete DKA100:[YOUNG]COMMIT_OR_ROLLBACK.SQL;*
$ if f$search("DKA100:[YOUNG]CONTINUE_OR_EXIT.SQL") .nes. "" then -
     delete DKA100:[YOUNG]CONTINUE_OR_EXIT.SQL;*
$ if "''HOLD_DELETE'" .nes. "" then -
     DELETE :== "''HOLD_DELETE'"
$ set default 'DEFAULT_DIR
$!
$EXIT:
$ exit
```

DBTune Reports

Four reports are created by the DBTune procedure and they are placed in your default directory. The four reports are:

1. <<database_id>>_REVIEW_AND_GUIDE.REPORT
2. <<database_id>>_ANALYSIS.REPORT
3. <<database_id>>_TUNING_DETAIL.REPORT
4. <<database_id>>_DBTUNE.LOG

where <<database_id>> is the first ten characters of the .RDB file name.

The **REVIEW_AND_GUIDE.REPORT** provides tuning advice, instructions for executing the transformation procedure, and any warnings about errors that occurred or requirements that must be met for the procedure to execute properly. It also includes recommendations for setting certain **VMS AUTHORIZE** or **SYSGEN** parameters that may need to be changed. It is STRONGLY suggested that this report be read prior to executing the transformation **MAIN_DRIVER** command file.

The **ANALYSIS** report provides a narrative of the database's tuning and complexity status prior to executing the DBTune transformation procedure.

The **TUNING_DETAIL** report lists values used or calculated during the tuning process. B-tree information is also included for storage areas containing a single sorted index.

The **DBTUNE.LOG** report is a log of the DBTune process, showing the tables and indexes read as well as the storage areas that were tuned. In addition, results or errors encountered while parsing the PAD and Disk Utilization data files can be found in this file.

Following is an example of each of these reports.

Review and Guide Report

```
************************************************************************
*
* FILE    : MF_PERSONN_REVIEW_AND_GUIDE.REPORT
* CREATED : 04/03/2006 10:48:25  (DBTune V6.0)
*
*  Database tuning results and advice, reminders of tuning para-
*  meters that were set, and findings from common error checking.
*  Also contains results from determining database changes
*  and from creation of SQL as well as instructions on how
*  to execute the command files to alter the database.
*
************************************************************************
AUTOSQL: Automatic SQL Creation process began at 04/03/2006 10:48:25
========================================================================

  Specifications gathered for database: MF_PERSONNEL
  Database to be tuned via SQL EXPORT/IMPORT.

========================================================================


Storage Area Page Size Distribution (For Tuned Areas):

       PAGE SIZE   # of SA's
       ---------   ---------
            2              1
            3              7
            4              5
            6              1
           12              1
       ---------   ---------
Most frequent       : 3
Next most frequent  : 4
Largest page size   : 12
Calc'd buffer size  : 12


              Cardinality Distribution for Tables and Their Indices
          Cardinality Range        #Tables   #Indices   Total   Accum
        --------------------------  -------  --------  -----  -----
        1,000,000 or more records:  00000      00000   000000  000000
        100,000 to 999,999       :  00000      00000   000000  000000
        50,000 to 99,999         :  00000      00000   000000  000000
        10,000 to 49,999         :  00000      00000   000000  000000
        5,000 to 9,999           :  00000      00000   000000  000000
        2,500 to 4,999           :  00000      00000   000000  000000
        1,000 to 2,499           :  00000      00000   000000  000000
        500 to 999               :  00001      00001   000002  000002
        250 to 499               :  00001      00002   000003  000005
        100 to 249               :  00002      00005   000007  000012
        0 to 99                  :  00006      00002   000008  000020
```

```
vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv
>                                                                            <
> -Number of storage areas successfully tuned     :    15                   <
>                                                                            <
> -Total number of storage areas in tuned database:    15                   <
>                                                                            <
> -Total number of tables/indices stored in the                              <
<  database system storage area (RDB$SYSTEM) - May                           <
<  include SQLNET items which are NOT tuned       :     8                    <
>                                                                            <
> For details of storage area tuning see TUNING DETAIL report.              <
>                                                                            <
vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv

DATABASE PARAMETER CHANGES:
--------------------------
   - Old buffer size = 6  New buffer size = 12
   - Old buffer count = 20  New buffer count = 50
   - Old global buffs = DISABLED  New global buffs = DISABLED
   - Old number users = 50  New number users = 50

   Because of tuning changes, certain AUTHORIZE and SYSGEN
   parameters may need to be changed for users or systems
   which access this database.

   AUTHORIZE parameter  Recommended MINIMUM settings
   -------------------  ------------------------------------------------------
        FILLM           130 (cannot exceed SYSGEN param 'CHANNELCNT')
        DIOLM           150
        BIOLM           150
        ASTLM           263
        ENQLM           16000
        BYTLM           64000
        WSDEF           2048
        WSQUO           4096
        WSEXTENT        8192 (cannot exceed SYSGEN param 'WSMAX')
        PGFLQUOTA       160000 (cannot exceed SYSGEN param 'VIRTUALPAGECNT')

    * How to calculate WORKING SET sizes:
      ----------------------------------
      1) $ SHOW MEM/PHYS
      2) AVAIL_MEM = 'TOTAL' - 'PERMANENTLY ALLOC TO VMS' - 3000
      3) Determine the MAX number of users on your system (online, batch, etc)
      4) WSQUO = AVAIL_MEM / MAX_USERS
      5) WSDEF = 2048, 4096, or same as WSQUO
      6) WSEXT = multiple of WSQUO; can go as high as AVAIL_MEM but
                 cannot exceed WSMAX (SYSGEN param)
```

```
INDEX ANALYSIS:
---------------

Following are the results of a physical index analysis for this database
which scans for elements in the index attributes in an attempt to highlight
possible improvements in index design:

================================================================================

* The following indices are SORTED and contain a keyword which typically
  represents a unique value.  Because hashed keys require an exact match and
  sorted keys are better suited for range retrievals, these indices may
  provide better performance as HASHED keys rather than as sorted keys...

  INDEX NAME                      ATTRIBUTE ID                    KEY WORD
  ------------------------------- ------------------------------- ----------
  COLL_COLLEGE_CODE               COLLEGE_CODE                    CODE
  DEG_COLLEGE_CODE                COLLEGE_CODE                    CODE
  DEG_EMP_ID                      EMPLOYEE_ID                     ID
  DEPARTMENTS_INDEX               DEPARTMENT_CODE                 CODE
  EMP_EMPLOYEE_ID                 EMPLOYEE_ID                     ID
  JH_EMPLOYEE_ID                  EMPLOYEE_ID                     ID
  SH_EMPLOYEE_ID                  EMPLOYEE_ID                     ID


Out of 10 indices scanned, 7 were found whose design could
potentially be improved.
================================================================================

Following are indices with an average number of duplicates that exceed 10.
Indices with many duplicates can seriously degrade I/O performance.
To reduce the number of duplicates for an index, additional key fields can be
added or two or more indices can be combined to form a larger index.
If an index with many duplicates is needed, a sorted ranked index may
reduce storage requirements and improve performance.

INDEX NAME                     AVERAGE DUPLICATES
==============================  ==================
DEG_COLLEGE_CODE                12
```

```
************************************************************
*                                                          *
* Following is a summary of the database changes found:    *
* -------------------------------------------------------- *
*  - Number of INDICES altered          :    10            *
*  - Number of STORAGE MAPS altered     :    10            *
*  - Number of STORAGE AREAS added      :     9            *
*  - Number of STORAGE AREAS altered    :     6            *
*  - Number of STORAGE AREAS dropped    :     4            *
*  - Number of DATABASE PARAMS specified:    13            *
*                                       ---------          *
*      TOTAL DATABASE CHANGES FOUND          52            *
*                                                          *
************************************************************


DATABASE TUNING SUMMARY:
------------------------

******************** Disk Space Requirements ***********************

 o Approx size of tuned database         :               8283 blocks
 o Approx size of the BACKUP file        :               2142 blocks
 o Approx size of the SQL EXPORT file    :                255 blocks

Before executing the MAIN DRIVER procedure, ensure that adequate
free space exists on the disks to which the database and its
associated files are assigned:

 o RDB root area  : DKA100:[TEST72]
 o RMU/BACKUP area: DKA100:[YOUNG]
 o SQL EXPORT area: DKA100:[YOUNG]
 o Review the MF_PERSONN.DISKUTIL file for disk requirements

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


TO EXECUTE THE GENERATED SQL SCRIPTS:
-------------------------------------

         All of the files listed below are located in
         the SQL directory: DKA100:[YOUNG]

         The scripts that are created are driven by...
             @MF_PERSONN_TRANSFORM.MAIN_DRIVER

         It executes the following command files:
             RMU/BACKUP of the database
             MF_PERSONN.ALTERS1_SQL
             MF_PERSONN.EXPORT_SQL
             DROP the database
             MF_PERSONN.IMPORT_SQL
             MF_PERSONN.ALTERS2_SQL

During execution of the MAIN_DRIVER command, a log file will be created
in the SQL directory and will be named MF_PERSONN.SQL_LOG.
This file can be viewed for SQL messages or errors that may have occurred
during execution of the scripts and can be deleted after viewing.

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
IMPORTANT:
     For the database transformation procedure to succeed, any
```

```
     VMS logicals which are required to invoke this database
     must be assigned PRIOR to the transformation!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

AUTOSQL: SQL Creation proccess ended successfully at 04/03/2006 10:48:30
```

Analysis Report

```
**********************************************************************
*
* FILE    : MF_PERSONN_ANALYSIS.REPORT
* CREATED : 04/03/2006 10:56:47
*
* Narrative analysis of current database characteristics
*
**********************************************************************

Database Complexity
-------------------
    The complexity rating is a weighted measure of the database
design and its stored records.  A rating of 8 indicates a
relatively small database.  Tuning requirements are simple.
    The largest factor in this rating is the domain count.
It accounts for 13% of the complexity rating.  The next largest
component of this rating is the column count, which is also 13%.
The complexity will increase as records and Rdb items (e.g.
tables, columns) are added.

Database Tune Rating
--------------------
    The tune rating is a composite measure of the physical storage
design as it applies to the logical structure of this database.
The complexity rating of 8 and the tune rating of 32 indicate
that the physical storage strategy should allow this database to
perform reasonably well and allow for some growth in complexity
without noticeable performance degradation.
    The tune rating measures significant factors that affect the
physical storage design.  It does not consider every factor, but
does objectively measure factors critical to successful Rdb tuning.
Remember that overall performance is a function of many things,
including system load, system tuning, and application design, in
addition to the physical storage strategy.

Database Storage Area Allocation
--------------------------------
    The 'Storage Area Allocation' data indicate the percentage of
allocated space that has been extended for both RDA and SNP files.
Of the total RDA pages, a large percentage (68%) have been
extended as these storage areas have been loaded.   RDA pages are
extended when the data requirements for tables and/or indices exceed
the existing allocation of pages.  The RDA areas have been extended
9 times.

    Of the total SNP pages, a large percentage (97%) have been
created as these storage areas have been utilized.  SNP files are
used to enable READ-only transactions to access data concurrently
while WRITE transactions are active.  SNP pages are only used if
SNAPSHOTS ARE ENABLED.

Database Index Analysis
-----------------------
    The index analysis data indicate that 20% of the database indices are HASHED.
Thus, 80% of the indices are SORTED.  DBTune reviewed the indices
and found that five of the SORTED indices are candidates to be HASHED
and none of the HASHED indices are candidates to be SORTED.  Review
```

```
the REVIEW and GUIDE report to see which indices have been selected.
    DBTune looks for certain key words within the index columns.  It
assumes certain types of queries will be made based on these key words.
The person responsible for maintaining the database should review
actual usage to determine whether to modify the index.
    NOTE:  HASHED indices facilitate exact match queries.
           They incur narrow locks for updates.
           SORTED indices facilitate sequential retrievals.
           They incur broader lock contention for updates
           than HASHED indices.
```

Tuning Detail Report

```
*************************************************************************
*
*  FILE    : MF_PERSONN_TUNING_DETAIL.REPORT
*  CREATED : 04/03/2006 10:48:25  (DBTune V6.0)
*
*  Detail of parameters and calculated values resulting from tuning
*  the storage areas for MF_PERSONNEL.
*  For tuning summaries and advice, see REVIEW & GUIDE report.
*
*************************************************************************
*
 NOTE:  For those storage areas containing a single SORTED index, B-tree
        information will be printed (tree levels, duplicate nodes, etc).
        Be aware that for a non-compressed index each B-tree
        is based on the record count, index fill percentage, growth rate,
        access bias, and node size listed for the particular index involved.
        If compression is specified for an index, its actual B-tree may be
        smaller than represented below.

 Values used to calculate page size for: DEG_COLLEGE_CODE_IDX
 ------------------------------------------------------------------------------
  * B-TREE LEVELS FOR INDEX: DEG_COLLEGE_CODE ...
      - B-Tree consists of only ONE node

    STORED ITEMS (1 sorted): DEG_COLLEGE_CODE
    SORT KEY SIZE/SEGMENTS : 4 / 1
    # OF RECS (10 % growth): 182
    NODE SIZE (bias : 50 ) : 560
    INDEX RECORDS PER NODE : 33  (29 with 90% fill)
    NODES PER PAGE ....... : 3
    AVERAGE DUPLICATES ... : 12
    PAGE SIZE/ALLOCATION   : 4 / 16
    TOTAL RDA BLOCKS ..... : 64
    TOTAL SNP BLOCKS ..... : 24

 Values used to calculate page size for: DEG_EMP_ID_IDX
 ------------------------------------------------------------------------------
  * B-TREE LEVELS FOR INDEX: DEG_EMP_ID ...
      - LEVEL   1 : 007 nodes  (BOTTOM of the tree)
      - LEVEL 002 : 001 node   (TOP of the tree)
=================================================
      TOTAL B-TREE NODES    : 8 nodes
      LEVELS IN THE TREE    : 2 levels

    STORED ITEMS (1 sorted): DEG_EMP_ID
    SORT KEY SIZE/SEGMENTS : 5 / 1
    # OF RECS (10 % growth): 182
    NODE SIZE (bias : 50 ) : 593
    INDEX RECORDS PER NODE : 33  (29 with 90% fill)
    NODES PER PAGE ....... : 3
    AVERAGE DUPLICATES ... : 1
    PAGE SIZE/ALLOCATION   : 4 / 13
    TOTAL RDA BLOCKS ..... : 52
    TOTAL SNP BLOCKS ..... : 24

 Values used to calculate page size for: EMPIDS_LOW
 ------------------------------------------------------------------------------
```

```
    N-CLUSTER ITEMS        : 2 table(s)
                           : 2 hash scattered index(es)
    TABLE (ByteCt / Cols)  : EMPLOYEES (112 / 12)
    TABLE (ByteCt / Cols)  : JOB_HISTORY (34 / 6)
    INDEX (KeySz / AvgDups): EMPLOYEES_HASH (5 / none)
    INDEX (KeySz / AvgDups): JOB_HISTORY_HASH (5 / 2)
    N-CLUSTER RECORD + OH  : 190
    SPAM THRESHOLDS ...... : 90, 91, 99
    PAGE SIZE/ALLOCATION   : 3 / 39
    TOTAL RDA BLOCKS ..... : 117
    TOTAL SNP BLOCKS ..... : 18

 Values used to calculate page size for: EMPIDS_MID
 ------------------------------------------------------------------------------
    N-CLUSTER ITEMS        : 2 table(s)
                           : 2 hash scattered index(es)
    TABLE (ByteCt / Cols)  : EMPLOYEES (112 / 12)
    TABLE (ByteCt / Cols)  : JOB_HISTORY (34 / 6)
    INDEX (KeySz / AvgDups): EMPLOYEES_HASH (5 / none)
    INDEX (KeySz / AvgDups): JOB_HISTORY_HASH (5 / 2)
    N-CLUSTER RECORD + OH  : 190
    SPAM THRESHOLDS ...... : 90, 91, 99
    PAGE SIZE/ALLOCATION   : 3 / 43
    TOTAL RDA BLOCKS ..... : 129
    TOTAL SNP BLOCKS ..... : 18

 Values used to calculate page size for: EMPIDS_OVER
 ------------------------------------------------------------------------------
    N-CLUSTER ITEMS        : 2 table(s)
                           : 2 hash scattered index(es)
    TABLE (ByteCt / Cols)  : EMPLOYEES (112 / 12)
    TABLE (ByteCt / Cols)  : JOB_HISTORY (34 / 6)
    INDEX (KeySz / AvgDups): EMPLOYEES_HASH (5 / none)
    INDEX (KeySz / AvgDups): JOB_HISTORY_HASH (5 / 2)
    N-CLUSTER RECORD + OH  : 190
    SPAM THRESHOLDS ...... : 90, 91, 99
    PAGE SIZE/ALLOCATION   : 3 / 32
    TOTAL RDA BLOCKS ..... : 96
    TOTAL SNP BLOCKS ..... : 18

 Values used to calculate page size for: EMP_EMPLOYEE_ID_IDX
 ------------------------------------------------------------------------------
  * B-TREE LEVELS FOR INDEX: EMP_EMPLOYEE_ID ...
      - LEVEL   1 : 004 nodes  (BOTTOM of the tree)
      - LEVEL 002 : 001 node   (TOP of the tree)
================================================
      TOTAL B-TREE NODES    : 5 nodes
      LEVELS IN THE TREE    : 2 levels

    STORED ITEMS (1 sorted): EMP_EMPLOYEE_ID
    SORT KEY SIZE/SEGMENTS : 5 / 1
    # OF RECS (10 % growth): 110
    NODE SIZE (bias : 50 ) : 593
    INDEX RECORDS PER NODE : 33  (29 with 90% fill)
    NODES PER PAGE ....... : 3
    AVERAGE DUPLICATES ... : 0
    PAGE SIZE/ALLOCATION   : 4 / 12
    TOTAL RDA BLOCKS ..... : 48
    TOTAL SNP BLOCKS ..... : 24

 Values used to calculate page size for: EMP_INFO
 ------------------------------------------------------------------------------
    STORED ITEMS (1 table) : DEGREES
```

```
    TABLE BYTE COUNT/SEGS  : 29 / 5
    COMPRESSION            : AvgLen+OH=29 bytes
    # OF RECS (10% growth) : 182
    TABLE/PARTITION RECORDS: 51 (per page)
    ACCESS BIAS            : 50
    PAGE SIZE/ALLOCATION   : 3 / 14
    SPAM THRESHOLDS        : 96, 98, 98
    TOTAL RDA BLOCKS       : 42
    TOTAL SNP BLOCKS       : 18

 Values used to calculate page size for: EMP_LAST_NAME_IDX
 ------------------------------------------------------------------------------
  * B-TREE LEVELS FOR INDEX: EMP_LAST_NAME ...
      - LEVEL   1 : 004 nodes  (BOTTOM of the tree)
      - LEVEL 002 : 001 node   (TOP of the tree)
================================================
      TOTAL B-TREE NODES    : 5 nodes
      LEVELS IN THE TREE    : 2 levels

    STORED ITEMS (1 sorted): EMP_LAST_NAME
    SORT KEY SIZE/SEGMENTS : 14 / 1
    # OF RECS (10 % growth): 110
    NODE SIZE (bias : 50 ) : 890
    INDEX RECORDS PER NODE : 33  (29 with 90% fill)
    NODES PER PAGE ....... : 3
    AVERAGE DUPLICATES ... : 1
    PAGE SIZE/ALLOCATION   : 6 / 12
    TOTAL RDA BLOCKS ..... : 72
    TOTAL SNP BLOCKS ..... : 36

 Values used to calculate page size for: JH_EMPLOYEE_ID_IDX
 ------------------------------------------------------------------------------
  * B-TREE LEVELS FOR INDEX: JH_EMPLOYEE_ID ...
      - LEVEL   1 : 006 nodes  (BOTTOM of the tree)
      - LEVEL 002 : 001 node   (TOP of the tree)
================================================
      TOTAL B-TREE NODES    : 7 nodes
      LEVELS IN THE TREE    : 2 levels
      LARGE DUPLICATE NODES : 16 nodes
      SMALL DUPLICATE NODES : 118 nodes

    STORED ITEMS (1 sorted): JH_EMPLOYEE_ID
    SORT KEY SIZE/SEGMENTS : 5 / 1
    # OF RECS (10 % growth): 301
    NODE SIZE (bias : 50 ) : 593
    INDEX RECORDS PER NODE : 33  (29 with 90% fill)
    NODES PER PAGE ....... : 3
    AVERAGE DUPLICATES ... : 2
    PAGE SIZE/ALLOCATION   : 4 / 26
    TOTAL RDA BLOCKS ..... : 104
    TOTAL SNP BLOCKS ..... : 24

 Values used to calculate page size for: LIST_AREA
 ------------------------------------------------------------------------------
    STORED ITEMS           : Default List Area (Segmented Strings)
    PAGE SIZE/ALLOCATION   : 12 / 44
    TOTAL RDA BLOCKS ..... : 528
    TOTAL SNP BLOCKS ..... : 72

 Values used to calculate page size for: RDB$SYSTEM
 ------------------------------------------------------------------------------
    STORED ITEMS           : 0 table(s)
                           : 0 sorted index(es)
```

```
                          : 0 hash scattered index(es)
    PAGE SIZE/ALLOCATION   : 2 / 1533
    TOTAL RDA BLOCKS ..... : 3066
    TOTAL SNP BLOCKS ..... : 154

 Values used to calculate page size for: SALARY_HISTORY
 ------------------------------------------------------------------------------
    STORED ITEMS (1 table) : SALARY_HISTORY
    TABLE BYTE COUNT/SEGS  : 25 / 4
    COMPRESSION            : AvgLen+OH=28 bytes
    # OF RECS (10% growth) : 802
    TABLE/PARTITION RECORDS: 53 (per page)
    ACCESS BIAS            : 50
    PAGE SIZE/ALLOCATION   : 3 / 26
    SPAM THRESHOLDS        : 97, 98, 99
    TOTAL RDA BLOCKS       : 78
    TOTAL SNP BLOCKS       : 18

 Values used to calculate page size for: SH_EMPLOYEE_ID_IDX
 ------------------------------------------------------------------------------
  * B-TREE LEVELS FOR INDEX: SH_EMPLOYEE_ID ...
      - LEVEL   1 : 004 nodes  (BOTTOM of the tree)
      - LEVEL 002 : 001 node   (TOP of the tree)
================================================
      TOTAL B-TREE NODES    : 5 nodes
      LEVELS IN THE TREE    : 2 levels
      LARGE DUPLICATE NODES : 38 nodes
      SMALL DUPLICATE NODES : 44 nodes

    STORED ITEMS (1 sorted): SH_EMPLOYEE_ID
    SORT KEY SIZE/SEGMENTS : 5 / 1
    # OF RECS (10 % growth): 802
    NODE SIZE (bias : 50 ) : 593
    INDEX RECORDS PER NODE : 33  (29 with 90% fill)
    NODES PER PAGE ....... : 3
    AVERAGE DUPLICATES ... : 7
    PAGE SIZE/ALLOCATION   : 4 / 28
    TOTAL RDA BLOCKS ..... : 112
    TOTAL SNP BLOCKS ..... : 24

 Values used to calculate page size for: SMALL_SORTED_AREA
 ------------------------------------------------------------------------------
    STORED ITEMS           : 0 table(s)
                           : 2 sorted index(es)
                           : 0 hash scattered index(es)
    SORTED (NodeSz/AvgDups): COLL_COLLEGE_CODE (560/none)
    SORTED (NodeSz/AvgDups): DEPARTMENTS_INDEX (560/none)
    LARGEST 'GROUP' + OH   : 570
    PAGE SIZE/ALLOCATION   : 3 / 19
    TOTAL RDA BLOCKS ..... : 57
    TOTAL SNP BLOCKS ..... : 18

 Values used to calculate page size for: SMALL_TABLE_AREA
 ------------------------------------------------------------------------------
    STORED ITEMS           : 6 table(s)
                           : 0 sorted index(es)
                           : 0 hash scattered index(es)
    TABLE (ComprRec / Cols): CANDIDATES (90 / 4)
    TABLE (ComprRec / Cols): COLLEGES (48 / 5)
    TABLE (ComprRec / Cols): DEPARTMENTS (46 / 5)
    TABLE (ComprRec / Cols): JOBS (34 / 5)
    TABLE (ComprRec / Cols): RESUMES (17 / 2)
    TABLE (ComprRec / Cols): WORK_STATUS (26 / 3)
```

```
    LARGEST 'GROUP' + OH   : 301
    PAGE SIZE/ALLOCATION   : 3 / 48
    TOTAL RDA BLOCKS ..... : 144
    TOTAL SNP BLOCKS ..... : 18
```

DBTune Process Log Report

```
********************************************************************************
*
* FILE    : MF_PERSONN_DBTUNE.LOG
* CREATED : 04/03/2006 10:48:04  (6.0)
*
* Log of the parameter settings and execution of the DBTune procedure for...
*
* Database: MF_PERSONNEL
*
********************************************************************************
*** Total BUFFER-PAGE records successfully parsed from ALI_BUFFER_PAGE: 29
*** Total BIAS-SCALE records successfully parsed from ALI_BIAS_SCALE: 101

Following are the parameter settings used for this execution of DBTune:
------------------------------------------------------------------------------
Rdb Version   = 7.2
Strategy      = N  --> (Create all NEW storage areas)
TuneTechnique = SQL  --> (Use SQL Export/Import to tune)
DBDisks       = 1  --> (See next 2 lines)
DBDISK001     = DKA100:[TEST72]
(DBDISK001    file types: /TBLRDA/TBLSNP/IDXRDA/IDXSNP/SYSRDA/SYSSNP/SYSRDB/)
Edit Files    = N  --> (Do NOT edit PAD & Diskutil during process)
ALI Editor    = EDIT/EDT
ModPAD file   = --> (No file specified)
DynWork file  = --> (No file specified)
Access Bias   = 50  --> (50% READ biased)
Index Fill %  = 90% --> (fill for sorted index nodes)
Growth %      = 10% --> (estimated database growth)
Snapshot %    = 5% --> (% of RDA allocated for SNP)
Min Page Size = 1  --> (Min storage area page size in blocks)
Max Page Size = 32  --> (Max storage area page size in blocks)
Min Buff Size = 6  --> (Min database buffer size in blocks)
Max Buff Size = 64  --> (Max database buffer size in blocks)
Min Buffers   = 20  --> (Minimum # of database buffers)
Max Buffers   = 100  --> (Maximum # of database buffers)
Sys Mem Pages = 0  --> (use existing db global buffer settings)
Max DB Users  = 0  --> (use existing db users setting)
Spread Areas  = B  --> (Based on BOTH Volume & Activity)
Logicals      = N  --> (No logs; use physical locations)
Logical Type  = PROCESS  --> (No logs; use physical locations)
Conceal Logs  = N  --> (No logs; use physical locations)
Load Time Lim = 0  --> (has no effect for SQL Export/Import)
Machine VUPs  = 2.5  --> (has no effect for SQL Export/Import)
Table Commit  = Y  --> (has no effect for SQL Export/Import)
Save Comments = Y  --> (has no effect for SQL Export/Import)
Tune for Comp = Y  --> (Use COMPRESSED data values for tuning)
Min Card      = 100  --> (See next section below)
-- Storage Areas for Items with a Cardinality Below: 100 ('Min Card') --
Tables        = SMALL_TABLE_AREA
Sorted        = SMALL_SORTED_AREA
Hashed        = SMALL_HASHED_AREA
```

```
------------------------  Assigned Directories-------------------------
SQL Dir       = DKA100:[YOUNG]
RUJ Dir       = DKA100:[YOUNG]
Backup Dir    = DKA100:[YOUNG]
ExpUnl Dir    = DKA100:[YOUNG]
--------------------------  Assigned Logicals  ---------------------------
ALI_RDB_DATABASE   = DKA100:[TEST72]MF_PERSONNEL.RDB
ALI_DBTUNE_PARAMS  = QUARK$DKA100:[DBT.VER60.DEV]DBTUNE_DEFAULT.PARAMS
ALI_DBTUNE_HOME    = ITA$1:[DBT.VER60.DEV]
ALI_DBTUNE_SCRATCH = ITA$1:[DBT.VER60.DEV.SCRATCH]
ALI_BUFFER_PAGE    = ALI_DBTUNE_HOME:BUFFER_PAGE.DAT
ALI_BIAS_SCALE     = ALI_DBTUNE_HOME:BIAS_SCALE.DAT
------------------------------------------------------------------------------

DBTUNE: Initializing Metadata for DBTune...

DBTUNE: Reading Rdb Tables...
        Reading table: CANDIDATES
        Reading table: COLLEGES
        Reading table: DEGREES
        Reading table: DEPARTMENTS
        Reading table: EMPLOYEES
        Reading table: JOBS
        Reading table: JOB_HISTORY
        Reading table: RESUMES
                      (^ a list table)
        Reading table: SALARY_HISTORY
        Reading table: WORK_STATUS

DBTUNE: Reading Rdb Indices...
        Reading index: COLL_COLLEGE_CODE
        Reading index: DEG_EMP_ID
        Reading index: DEG_COLLEGE_CODE
        Reading index: DEPARTMENTS_INDEX
        Reading index: EMP_LAST_NAME
        Reading index: EMP_EMPLOYEE_ID
        Reading index: EMPLOYEES_HASH
        Reading index: JH_EMPLOYEE_ID
        Reading index: JOB_HISTORY_HASH
        Reading index: SH_EMPLOYEE_ID

DBTUNE: Scanning table areas for compression...

DBTUNE: Scanning (1) DEPARTMENTS
DBTUNE: Scanning (2) EMPIDS_LOW
DBTUNE: Scanning (3) EMPIDS_MID
DBTUNE: Scanning (4) EMPIDS_OVER
DBTUNE: Scanning (5) EMP_INFO
DBTUNE: Scanning (6) JOBS
DBTUNE: Scanning (7) RDB$SYSTEM
DBTUNE: Scanning (8) RESUMES
DBTUNE: Scanning (9) RESUME_LISTS
DBTUNE: Scanning (10) SALARY_HISTORY

DBTUNE: Searching for N-clustered items...
        - The EMPLOYEES table is a member of
          the N-cluster area: EMPIDS_LOW
                              EMPIDS_MID
                              EMPIDS_OVER
        - The EMPLOYEES_HASH index is a member of
          the N-cluster area: EMPIDS_LOW
                              EMPIDS_MID
                              EMPIDS_OVER
```

```
        - The JOB_HISTORY table is a member of
          the N-cluster area: EMPIDS_LOW
                              EMPIDS_MID
                              EMPIDS_OVER
        - The JOB_HISTORY_HASH index is a member of
          the N-cluster area: EMPIDS_LOW
                              EMPIDS_MID
                              EMPIDS_OVER

DBTUNE: Gathering information for the tuning process...

DBTUNE: Tuning scripts for database storage areas...

Tuning script for area: DEG_COLLEGE_CODE_IDX
Tuning script for area: DEG_EMP_ID_IDX
Tuning script for area: EMPIDS_LOW
Tuning script for area: EMPIDS_MID
Tuning script for area: EMPIDS_OVER
Tuning script for area: EMP_EMPLOYEE_ID_IDX
Tuning script for area: EMP_INFO
Tuning script for area: EMP_LAST_NAME_IDX
Tuning script for area: JH_EMPLOYEE_ID_IDX
Tuning script for area: LIST_AREA
Tuning script for area: RDB$SYSTEM
Tuning script for area: SALARY_HISTORY
Tuning script for area: SH_EMPLOYEE_ID_IDX
Tuning script for area: SMALL_SORTED_AREA
Tuning script for area: SMALL_TABLE_AREA

Storage Areas Ordered by Weight (Weighted by Volume & Activity)...
-------------------------------------------------------------------------
RDB$SYSTEM                      --> WGT = 999999999
EMPIDS_OVER                     --> WGT = 86
EMPIDS_MID                      --> WGT = 86
EMPIDS_LOW                      --> WGT = 86
SH_EMPLOYEE_ID_IDX              --> WGT = 81
SALARY_HISTORY                  --> WGT = 81
JH_EMPLOYEE_ID_IDX              --> WGT = 31
EMP_INFO                        --> WGT = 19
DEG_EMP_ID_IDX                  --> WGT = 19
DEG_COLLEGE_CODE_IDX            --> WGT = 19
EMP_LAST_NAME_IDX               --> WGT = 12
EMP_EMPLOYEE_ID_IDX             --> WGT = 12
SMALL_TABLE_AREA                --> WGT = 10
SMALL_SORTED_AREA               --> WGT = 5
LIST_AREA                       --> WGT = 0

DBTUNE: Tuning script process SUCCESSFUL.

DBTUNE: Spreading storage areas over available disks...
Cost of placing 'DEG_COLLEGE_CODE_IDX' RDA file on disk 'DISK001' was 0
Cost of placing 'DEG_EMP_ID_IDX' RDA file on disk 'DISK001' was 0
Cost of placing 'EMPIDS_LOW' RDA file on disk 'DISK001' was 0
Cost of placing 'EMPIDS_MID' RDA file on disk 'DISK001' was 0
Cost of placing 'EMPIDS_OVER' RDA file on disk 'DISK001' was 0
Cost of placing 'EMP_EMPLOYEE_ID_IDX' RDA file on disk 'DISK001' was 0
Cost of placing 'EMP_INFO' RDA file on disk 'DISK001' was 0
Cost of placing 'EMP_LAST_NAME_IDX' RDA file on disk 'DISK001' was 0
Cost of placing 'JH_EMPLOYEE_ID_IDX' RDA file on disk 'DISK001' was 0
Cost of placing 'LIST_AREA' RDA file on disk 'DISK001' was 0
Cost of placing 'RDB$SYSTEM' RDA file on disk 'DISK001' was 0
Cost of placing 'SALARY_HISTORY' RDA file on disk 'DISK001' was 0
Cost of placing 'SH_EMPLOYEE_ID_IDX' RDA file on disk 'DISK001' was 0
```

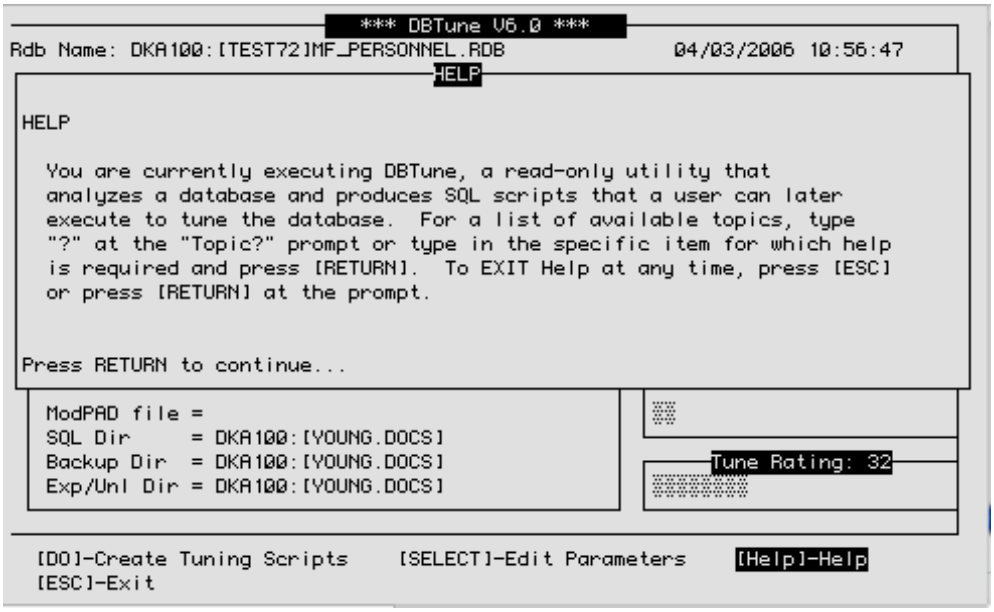
```
Cost of placing 'SMALL_SORTED_AREA' RDA file on disk 'DISK001' was 0
Cost of placing 'SMALL_TABLE_AREA' RDA file on disk 'DISK001' was 0

DBTUNE: Creating Disk Utilization data file

DBTUNE: Interpreting Disk Utilization data file

DBTUNE: Interpreting Disk Utilization data file

DBTUNE: Determining file specifications for storage areas

DBTUNE: Scanning database for modifications...
DBTUNE:   - Storage map changes
DBTUNE:   - Index changes (node size, type)
DBTUNE:   - Storage area changes
DBTUNE:   - Sequencing database changes
DBTUNE: Modification scan SUCCESSFUL.

DBTUNE: Generating transformation procedure...
DBTUNE:   - Generating transformation MAIN_DRIVER
DBTUNE:   - Generating EXPORT/IMPORT commands
DBTUNE:   - Generating SQL optimization scripts
DBTUNE: Generation of transformation procedure SUCCESSFUL.
```

DBTune Help

Online HELP is available within DBTune by pressing the **HELP** key or by selecting the **HELP** option in the DBTune menu and pressing **Return**. To obtain DBTune help outside of the DBTune utility, type the following command at the **DCL** prompt after installation of DBTune:

```
$ HELP/LIBRARY=ALI_DBTUNE_HOME:DBTUNE.HLB
```

Following is an example of the DBTune HELP window:



Appendix A

Customer Support

If you experience problems installing or using Rdb Controller for Rdb, please contact Customer Support at ALI.

How to Contact Customer Support:

Telephone:	(866) 257-8970 (803) 648-5931
Fax:	(803) 641-0345
Web Address:	www.aliconsultants.com
Support E-Mail:	support@aliconsultants.com

International clients may also obtain support through their local distributor's office.

Appendix B

Release Notes

Release notes and major changes since the last release may be found on the VMS CD in the file VMS_RELEASE_NOTES.TXT.

Appendix C

Glossary of Technical Terms

This appendix provides definitions of database terms and network terms that you may encounter in this manual or that may generally apply to the use of our products.

Database Terms

Term	Description
AIJ	In Rdb, image journaling stores copies of database rows after they are updated and committed. Records are stored in one or more AIJ files. After a system failure, AIJ can be used to reconstruct a database to include the last successfully completed transaction.
Availability	A measure of the percentage of time an application is up or down, or the percentage of time the application must be up and running (0 - 100 percent).
Cluster	Tables that are frequently accessed together may be physically stored together. To store them together, a *cluster* is created to hold the tables. The data in the tables is then stored together to minimize the number of I/Os that must be performed and thus improve performance.

Data (DML) Locks	The maximum number of DML locks is one for each table modified in a transaction. The value should equal the grand total of locks on tables referenced by all users.
Data Buffer Cache	The portion of reserved database memory that stores copies of physical data blocks. By holding data in the Data Buffer Cache, data can be more speedily accessed than from physical storage.
Data Buffer Hit Ratio	Determined by dividing the number of cache hits (logical reads that require no physical reads) by the number of logical reads and multiplying by one hundred. The resulting percentage, when monitored over an extended period of time, is a valuable indication of how successful a database is in maintaining frequently accessed data in memory.
Data File	An Oracle tablespace is comprised of one or more physical files. The data files associated with a tablespace store a database's data in that tablespace.
Database File	Physical files that contain all database information. For Oracle, this would include control files, redo log files, and data files. For Sybaseand SQL Server, this would include devices. These files can be manipulated by the operating system.
Database File Multiblock Read Count	Used to multi-block I/O. This is the maximum number of blocks readable in one I/O operation during a sequential scan. Values in the range of 4 to 32 are reasonable. The actual maximums are operating system specific. This term does not apply to Rdb.
Data Source	These are the data conduits between applications and Empirical Director. As you define applications in Empirical Director, you indicate the data source for each application.
DB Block Buffers	This is an INIT.ORA parameter for Oracle that determines the number of database blocks cached in memory (one buffer equals one block). Because this parameter affects how frequently a block is stored in memory, it has a significant impact on Oracle database performance. Increasing the value for this parameter will increase the likelihood that data will be stored in memory (consequently increasing the Data Buffer Cache Hit Ratio), but at the expense of greater memory consumption.

DB Block Size	The number of bytes per database block. Typical values are 2K (2048 bytes) and 4K (4096 bytes). This value typically does not change after being set at the time of database creation.
Device	A Sybase or SQL Server database is comprised of one or more physical files called devices. The database engine uses devices to store database data.
Dictionary (DDL) Lock	Protects the definition of a schema object while that object is being accessed by an ongoing DDL transaction. Whereas a DDL lock is automatically acquired by the database during any DDL operation, users cannot explicitly request DDL locks. DDL Locks fall into the following categories: exclusive, shared, and breakable parse (for Oracle), exclusive, shared, intent, update, and demand (for Sybase and SQL Server).
Dictionary Cache	Used by Oracle during database operation to ascertain that objects exist and that users have accessed them properly. Oracle also updates the Data Dictionary continuously to reflect changes in database structures, auditing, granting, and data. Data Dictionary Cache is the part of the Data Dictionary cached in SGA using the LRU (least recently used) algorithm for fast access.
Dictionary Cache Read Count	Related to the following statistics stored in the V$ROWCACHE table: GETS shows the total number of requests for information on the corresponding items; GETMISSES shows the number of data requests resulting in cache misses. For frequently accessed Dictionary Caches, the Dictionary Cache Hit Ratio should be higher than 90 percent. To increase the Dictionary Cache Hit Ratio, increase the value of the initialization parameter SHARED_POOL_SIZE. This increases the memory available to the data dictionary cache. Used by Oracle.
Distributed Lock	Used by Oracle to ensure that the data and other resources distributed among the various instances of an Oracle Parallel Server remain consistent.
Execution (Explain) Plan	To execute a DML statement, a database may have to perform many steps. Each of these steps either physically retrieves rows of data from the database or prepares them in some way for the user issuing the statement.

Export An Oracle utility used to write data from an Oracle instance to the file system. These data, stored as export dump files, contain schema definitions and their contents that can be later imported back into the database. In Rdb, export is a SQL command rather than a separate utility.

FreeLists Used by Oracle to point to the available free blocks. The parameter, FreeLists, is used to specify the number of freelists for a segment. This parameter is particularly important when a large number of inserts have to be carried out in parallel. In this case the freelists parameter should be set to be greater than one.

Import An Oracle utility used to read data from an Oracle dump file on the file system into an Oracle instance. In Rdb, import is a SQL command rather than a separate utility.

Index An object created on a table to increase performance of data retrieval from the indexed table. The index is logically and physically independent of the table data.

Initial Used by Oracle to set the size of the first extent of a segment. This storage area is allocated when the segment is first created, and additional extents are added on later as needed.

Latch (Internal Lock) In Oracle, a simple low-level serialization mechanism to protect shared data structures in the SGA.

Library Cache In Oracle, the library cache contains shared SQL and PL/SQL areas. The Library Cache is contained in the shared pool (Shared SQL Area) within the SGA and is available to multiple, concurrent users.

Library Cache Hit Ratio In Oracle, the Library Cache Hit Ratio is the ratio of shared SQL and PL/SQL items found in the Library Cache versus physical storage. Related to the following statistics stored in the V$LIBRARYCACHE table: PINS, which shows the number of times an item in the library cache was executed; and RELOADS, which shows the number of library cache misses on execution steps.

Locking Mechanism	A database automatically locks a resource on behalf of a transaction to prevent other transactions from performing a task that requires exclusive access to the same resource. The lock is automatically released when certain events occur and the transaction no longer requires the resource.
MaxExtents	Used by Oracle to specify the maximum number of extents that can be allocated for a single segment.
MinExtents	Used by Oracle to specify the number of extents that should be allocated when the segment is first created**.** If minextents = 3, for example, the initial extent plus two times the next extent are allocated when the segment is first created.
Next	Used by Oracle to specify the size of the extents to be created after the first (initial) extent. This area is not allocated until necessary.
Optimal	Used by Oracle to specify the optimum size for a rollback segment. This parameter can only be set for rollback segments.
Parallel Cache Management (PCM) Lock	A distributed lock used by Oracle that covers one or more data blocks (table and index blocks) in the buffer cache.
PctFree	Sets the percentage of free space within a data block that will be reserved for possible updates to rows already stored within each data block of the segment. This term does not apply to Rdb.
PctIncrease	The percentage by which each incremental extent of a segment grows over the previous incremental extent allocated. This term does not apply to Rdb.
PctUsed	Sets the percentage of free space within a data block that must be available before row insertion into the data block will be allowed. This term does not apply to Rdb.
Pipes	These are the data conduits between applications and Empirical Director, also commonly referred to as “data sources.” As you define applications in Empirical Director, you indicate the data source for each application.

Redo Log File	Contains committed transactions that have not yet been written to the respective data files. This term does not apply to Rdb.
Referential Integrity	A rule defined on a column or set of columns in one table that allows you to insert or update a row only is the value for that column or set of columns (in the child table) matches the value in a column of a related table (parent table).
Response Time	The average time required to perform a transaction. Slow response time may be attributed to system-wide bottlenecks or to individual application problems.
Rollback Segment	A logical structure used by Oracle responsible for undoing uncommitted transactions.
RUJ	In Rdb, the recovery unit journaling stores copies of database rows before they are updated. The RUJ files are used to undo uncommitted updates to a database when a rollback is performed or a system failure occurs.
SGA Size	This is an Oracle parameter that sets the size of main memory allocated for exclusive use by Oracle.
SQL	SQL is short for Structured Query Language. This is a specialized programming language for sending queries to databases.
Sysop	Sysop means System Operator. The Sysop is anyone responsible for the physical operations of a computer system or network resource. A System Administrator decides how often backups and maintenance should be performed and the System Operator performs those tasks.
Table	The basic unit of data storage in a database. Within a table, data is stored in rows and columns. Each column is given a column name, a datatype, and a width and precision or scale. A row is a collection of column information corresponding to a single record.
Table Scans (Long Tables)	The total number of full table scans performed on tables with more than 5 db_blocks. When the number of full table scans is greater than 0 per transaction, the SQL statements in the application would benefit from tuning. This term does not apply to Rdb.

Table Scans (Short Tables)	The number of full table scans performed on tables with less than 5 db_blocks. It is optimal to perform full table scans on short tables rather than using indexes. This term does not apply to Rdb.
Tablespace	Logical storage space for database objects, such as tables, indexes, clusters, etc. Physically, a tablespace consists of one or more database files. Using multiple tablespaces allows for the logical separation of user data. This term does not apply to Rdb.
Throughput	The number of transactions per minute (0- 1,000,000) that the system is handling. This "workload" monitoring may include information on the number and type of transactions being executed, who is executing them, and the applications being used.
Transaction Efficiency	A ratio of the total number of transactions compared with the number of transactions that complete successfully. Also expressed as the percentage of successful database transactions (0 - 100 percent).
Trigger	A procedure that is implicitly executed when an INSERT, UPDATE, OR DELETE statement is issued against the associated table.

Network Terms

Term	Description
Backbone	Usually a high-performance network cable, such as fiber optic or thick wire, that provides the attachment point for branching network segments. For instance, a backbone may run completely through a building while hubs are attached at different points.
Bandwidth	Bandwidth refers to how much information can be sent through a connection. Usually bandwidth is measured in bits-per-second. A full page of English text is about 16,000 bits.
Baud	In common usage the baud rate of a modem is how many bits it can send or receive per second.

Bit An abbreviation of Binary DigIT. A bit is the smallest unit of computerized data. Bandwidth is usually measured in bits-per-second.

CGI CGI means Common Gateway Interface. CGI is a set of rules that describe how a Web server communicates with another piece of software on the same machine, and how the other piece of software (the CGI program) talks to the Web server. Usually a CGI program takes data from a Web server and does something with it, like putting the content of a form into an e-mail message, or turning the data into a database query.

Client Any device in a network that uses services from a server or servers. A PC is the most typical client on a network. Client may also refer to a software program that is used to contact and obtain data from a Server software program on another computer, usually across a great distance. Each Client program is designed to work with one or more specific kinds of Server programs, and each Server requires a specific kind of Client. (Also see "Server.")

Ethernet Ethernet is the most popular network medium in use today and is most popular in VAX, UNIX, and PC networks. It is a common method of networking computers in a LAN.

Hub Typically a device used to connect network cables. Twisted-pair systems use hubs to connect multiple cable segments to a backbone or other cable segment.

LAN Local Area Network. Network connecting systems that are usually in a single department, single building, or group of buildings.

MIB Modules MIB modules usually contain object definitions, may contain definitions of notifications, and sometimes include compliance statements specified in terms of appropriate object groups. MIB modules define the management information maintained by the instrumentation in managed nodes, made remotely accessible by management agents, conveyed by the management protocol, and manipulated by management applications. In general, management information defined in any MIB module, regardless of the version of the data definition language, can be used with any version of the protocol.

SNMP SNMP stands for Simple Network Management Protocol, a set of network communication specifications that cover the basics of network management in a method that poses little stress on the existing network. SNMP is a set of standards for communication with devices connected to a TCP/IP. Empirical Director uses SNMP to collect performance data.

Server A computer, or a software package, that provides a specific kind of service to client software running on other computers. The term can refer to a particular piece of software or to the machine on which the software is running. Any device that provides services to a network is considered to be a server. This may include file and print servers (such as UNIX, Novell NetWare, Windows NT Server, or VMS), terminal servers, application servers, and other specialized devices. (Also see "Client.")

TCP/IP TCP/IP is short for Transmission Control Protocol/Internet Protocol. This is the suite of protocols that defines the Internet.

Telnet The command and program used to login from one Internet site to another. The telnet command/program gets you to the login: prompt of another host.

Token Ring A newer network medium (relative to Ethernet) that is predominantly used in IBM shops.

Topology The physical architecture of the network, including cabling and configuration (such as a star, bus, or ring). Topologies have blurred during the last few years because most networks now consist of multiple topologies and cable types.

UNIX A computer operating system (the basic software running on a computer, underneath things like word processors and spreadsheets). UNIX is designed to be used by many people at the same time and has TCP/IP built in. It is the most common operating system for servers on the Internet.

WAN Wide Area Network. A network that usually interconnects LANs in different buildings or areas. For instance, a LAN in Building A connected to a LAN in Building B may be called a WAN.

Index