

Oracle 10g New Features for Administrators (Summary Sheets) v. 1.0

Installation, Server Configuration, and Database Upgrades

About Grid Computing

The following three attributes lie at the heart of grid computing:

- *Virtualization* between the layers of the computing stack and the users
- *Dynamic provisioning* of work among the available resources, based on changing needs
- *Pooling* of resources to maximize availability and utilization

Installation New Features Support

Database Management Choices

- You can manage your databases locally using the OEM Database Control, which is part of the Oracle 10g server software.
- You can manage your databases centrally, through the OEM Grid Control, which is available on separate CDs.

The Grid Control includes:

- Oracle Management Agent
- Oracle Management Service
- Oracle Management Repository
- Grid Control console

If you create a database manually, you must configure and install the OEM Database Control using the Oracle-supplied build script (EM Configuration Assistant):

- `$ORACLE_HOME/bin/emca` for UNIX
- `$ORACLE_HOME\bin\emca.bat` for Windows.

Note: In order to access the OEM Database Control from your browser, you must first have the `dbconsole` process running on your system.

Automatic Pre-Install Checks

Oracle Universal Installer (OUI) now manages the entire pre-install requirements check automatically. Common checks performed are the following:

- Correct operating system version and compatibility level
- Operating system patches
- Kernel parameters
- Sufficient memory and file space
- Oracle Home

New File Storage Options

The OUI now offers three choices for configuring the file systems for any new starter database that you may create:

- **Automatic Storage Management (ASM):** ASM is integration of a traditional file system with a built-in Logical Volume Manager (LVM). the database automatically stripes and mirrors your data across the available disks in the disk groups.

- **Raw Devices:** If you use RAC, and a Clustered File System (CFS) is available on your operating system, Oracle recommends using either CFS or ASM for your file storage. If a CFS is unavailable, Oracle recommends that you use raw, or "uncooked," file systems or ASM.
- **File Systems:** Choosing this option will mean that you are using the traditional operating system files and directories for your database storage.

Backup and Recovery Options

- Do not enable automatic backups
- Enable automatic backups

Database User Password Specification

You have to set passwords for the following schemas: SYS, SYSTEM, DBSNMP, and SYSMAN.

It's DBA job to unlock the other standard user accounts and set new passwords for them.

Cluster Ready Services

The Oracle 10g installation supports several Real Application Clusters (RAC) features, including the installation of the Cluster Ready Services (CRS) feature.

MetaLink Integration

In Oracle 10g, you can directly link the OEM to the OracleMetaLink service. Through this built-in MetaLink integration, OEM can then automatically track any new software patches for you. You can arrange to receive alerts whenever the OEM spots new patches.

Oracle Software Cloning

The OEM Grid Control enables you to easily duplicate Oracle Database 10g software installations (Oracle Homes) from a master installation to one more servers.

Database Cloning

Using the OEM, you can now easily clone databases. OEM performs database cloning by using RMAN. You use the OEM Clone Database wizard, also known as the Clone Database Tool, to perform the various steps in a database cloning operation.

Performance Enhancements to the Installation Process

Single CD Installation

Although the Oracle Database 10g server software comes in a pack of CD-ROMs, you need only a single CD to complete your Oracle 10g server installation. It takes only about 20 minutes to complete the entire installation.

Hardware Requirements

- **Memory:** You need 256MB for the basic database, and 512MB if you are using the stand-alone version of the OEM (the OEM Database Control).
- **Disk space:** You need a maximum of about 2.5GB of disk space for the Oracle software. In addition, you need 1GB of swap space and about 400MB of disk space in the /tmp directory.

Easier and Cleaner Deinstallation

In the deinstallation process, related software files and Widows registry entries are removed.

To deinstall your Oracle 10g software, follow these steps:

1. Shut down all databases and ASM instances running under the Oracle Home you want to remove, and then remove the databases.
2. Stop all the relevant processes running under this Oracle Home, by running the following commands:
`$ORACLE_HOME/bin/emctl stop dbconsole` - shuts down the OEM.
`$ORACLE_HOME/bin/lsnrctl stop` - brings down the Oracle listener
`$ORACLE_HOME/bin/isqlplusctl stop` - brings down the iSQL*Plus server
3. Start the OUI.
4. Click Deinstall Products in the Welcome window.
5. In the Inventory window, select the correct Oracle Home that contains the software you want to deinstall, and then click Remove.
6. Manually remove the Home directory that you just deinstalled.

Automatic Launching of Software

The following products will launch automatically immediately after you complete the server installation: Oracle Management Agent, the OEM Database Control, and the iSQL*Plus server.

Response File Improvements

The following are the new Oracle 10g improvements in the response file, which help you perform a truly "silent" Oracle installation:

- The file has a new header format, which makes the response file easier to edit.
- You don't need to specify an X server when performing installations in a character mode console.
- You don't need to set the DISPLAY variable on UNIX systems.
- No GUI classes are instantiated, making this a truly silent method of installing software.

Simplified Instance Configuration

Database Configuration Assistant (DBCA) Enhancements

Using the DBCA ensures that DBA is reminded about all the important options, rather than needing to remember them and perform them all manually. Following are some of the DBCA enhancements:

1. **The SYSAUX Tablespace:** This is a new tablespace introduced in Oracle 10g used as a central location for the metadata of all tools like the OEM and RMAN.
2. **Flash Recovery Area:** This is a unified storage location on your server that Oracle reserves exclusively for all database recovery-related files and activities.
3. **Automatic Storage Management (ASM)**
4. **Management Options:** like alert notification, job scheduling, and software management.

Policy-Based Database Configuration Framework

Oracle 10g enables you to monitor all of your databases to see if there are any violations of the predetermined configuration policies. This can be managed in the Database Control using following sections:

- Diagnostic Summary: shows you if there are any policy violations anywhere
- Policy Violations: summarizes all policy violations in your databases and hosts.
- Manage Policy Library: to disable any policy.

Simplified Initialization Parameters

- **Basic initialization parameters:** This set consists of about 25 to 30 of the most common parameters that you need for an Oracle database.
- **Advanced initialization parameters:** These are parameters you'll need to deploy only rarely, to improve your database's performance or to overcome some special performance problems.

Changes in the Initialization Parameters

Deprecated Parameters

MTS_DISPATCHERS
UNDO_SUPPRESS_ERRORS
PARALLEL_AUTOMATIC_TUNING

Obsolete Parameters

DISTRIBUTED_TRANSACTIONS
ORACLE_TRACE_COLLECTION_NAME
MAX_ENABLED_ROLES

New Parameters

RESUMABLE_TIMEOUT
SGA_TARGET
PLSQL_OPTIMIZE_LEVEL

Irreversible Datafile Compatibility

The minimum value of the COMPATIBLE initialization parameter is 9.2.0. The default value, however, is 10.0.0. If value of the parameter was 10.0.0, this means that you won't be able to downgrade the Oracle 10g database to a prior release; the datafile is *irreversible*.

The `alter database reset compatibility` command is obsolete in Oracle 10g.

Viewing Database Feature Usage Statistics

The Statistics Collection Process

Oracle Database 10g introduces a new database process called Manageability Monitor Process (MMON), which records both the database usage statistics and the HWM statistics for various objects.

MMON process is primarily responsible for:

- issuing database alerts
- collecting statistics
- taking snapshots of data into disks

MMON records the various statistics inside the Automatic Workload Repository (AWR), which is a new Oracle Database 10g innovation that stores database performance data.

The related views are:

- DBA_FEATURE_USAGE_STATISTICS to find out the usage statistics of various features that MMON has stored in the AWR.
- DBA_HIGH_WATER_MARK_STATISTICS to see the HWM statistics and a description of all the database attributes that the database is currently monitoring.

Database Usage Statistics in the OEM

Following are the steps to view database usage statistics in the OEM Database Control:

1. Go the Database Control home page. Click the **Administration** link and go to the **Configuration Management** group. Click the **Database Usage Statistics** link.

Supported Upgrade Paths to Oracle 10g

You can migrate directly to the Oracle Database 10g version only if your database is one of the following versions: 8.0.6, 8.1.7, 9.0.1, or 9.2.

You can upgrade to Oracle Database 10g in two ways:

- the traditional manual mode
- by using the Database Upgrade Assistant (DBUA)

Note: The DBUA is a GUI tool, but you can also run it in the silent mode, by using the following command at the operating system level: `dbua`

Using New Utility to Perform Pre-Upgrade Validation Checks

Oracle now includes a brand-new tool, called the **Upgrade Information Tool**, to help you collect various pieces of critical information before you start the upgrade process.

The Upgrade Information Tool provides important information and actions you should do before upgrading the existing database.

If you are performing a manual upgrade, you need to invoke the tool by running the SQL script `utlu101i.sql`. The DBCA automatically runs it as part of the pre-upgrade check.

The Post-Upgrade Status Tool

Oracle Database 10g also provides a Post-Upgrade Status Tool (`utlu101s.sql`), which gives you an accurate summary of the upgrade process and any necessary corrective steps to be taken.

You can restart a failed database upgrade job from the point where you failed.

If you use the DBUA to upgrade, the script runs automatically. If you are performing a manual upgrade, you need to run the script yourself, after the upgrade process is finished.

Using the Simplified Upgrade Process

Oracle provides the DBUA to facilitate the database upgrade process. You can use the DBUA to upgrade any database configuration, including RAC and standby databases.

The DBUA takes care of the following tasks for you:

- Deletes all obsolete initialization parameters
- Changes the `ORACLE_HOME` settings automatically
- Runs the appropriate upgrade scripts for your current release
- Configures your `listener.ora` file

Starting DBUA

On Windows: Programs | Oracle | Configuration and Migration Tools | Database Upgrade Assistant.

On a UNIX system: simply type `dbua`

Silent startup: `dbua -silent -dbName nina`

Manual Upgrade Process

Steps in the Manual Upgrade Process

1. Start a Spool File

```
SQL> spool upgrade.log
```

2. Run the Upgrade Information Tool

```
SQL> @$ORACLE_HOME/rdbms/admin/utlu101i.sql
SQL> spool off
```

3. Back Up Your Database

At this point, shut down and back up your current database, by using either the RMAN or by using user-managed backup techniques.

4. Copy Your init.ora File

Copy your present `init.ora` file to the new Oracle Database 10g default location:

- o `%ORACLE_HOME%\database` on Windows with the name: `init%ORACLE_SID%.ora`
- o `$ORACLE_HOME/dbs` under UNIX with the name: `init$ORACLE_SID.ora`

Make all the necessary changes in your `init.ora` parameter file, as per the Upgrade Information Tool's recommendations.

5. Completely Remove Any Windows-Based Oracle Instances

```
C:\>net stop oracleservicefinance
C:\>oradim -delete -sid finance
C:\>oradim -new -sid finance -intpwd finance1
-startmode auto -pfile
c:\oracle\product\10.1.0\Db_1\database\initfi
nance.ora
```

6. Start Up the New Database

```
C:\> sqlplus /nolog
SQL> connect / as sysdba
SQL> startup upgrade
```

Using the startup upgrade command tells Oracle to automatically modify certain parameters, including initialization parameters that cause errors otherwise

7. Create the SYSAUX Tablespace

```
CREATE TABLESPACE sysaux DATAFILE
'sysaux01.dbf' SIZE 500M
EXTENT MANAGEMENT LOCAL
SEGMENT SPACE MANAGEMENT AUTO
ONLINE;
```

8. Run the Upgrade Script

Run the Upgrade Script corresponding to the Oracle version you would like to upgrade:

- o 8.0.6: `u0800060.sql`
- o 8.1.7: `u0801070.sql`
- o 9.0.1: `u0900010.sql`
- o 9.2: `u0902000.sql`

9. Verify the Existence of Invalid Objects

```
select count(*) from dba_objects
where status = 'INVALID'
```

10. Recompile and Validate the Invalidated PL/SQL and Java Code

Run the script `utlrlp.sql`

11. Run the Post-Upgrade Status Tool

```
SQL> @utlu101s.sql TEXT
```

Note that the `utlu101s.sql` script is followed by the word `TEXT`, to enable the printing of the script output.

The tool simply queries the `DBA_SERVER_REGISTRY` table to determine the upgrade status of each individual component.

Reverting Upgraded Database

Instructing DBUA to perform a backup of your database (with the RMAN) will provide you the option to revert the database to the older version by the end of the upgrade process.

You can also revert back manually to the older database by using the `DB_Name_restore.bat` file (under Windows), providing that you have a cold backup of the database.

Loading and Unloading Data

Introduction to the Data Pump Architecture

Using Export and Import Data Pump utilities you can:

- export and import data faster than Old export/import utilities
- estimate job times
- perform fine-grained object selection
- monitor jobs effectively
- directly load one database from a remote instance
- call the utilities from PL/SQL using Data Dump API
- stop, resume and restart the utilities
- attach a running job to monitor jobs, as well as to modify certain parameters interactively.
- have fine-grained data import capability
- remap objects of a specific schema to another schema

Note : the export Data Pump user process launches a server-side process that writes data to disks on the server node, not the client that launches the utility.

Note: The new Data Pump technology lets you export data only to disk. You cannot use a tape drive when performing a Data Pump export.

Data Pump Components

- **The DBMS_DATAPUMP package**: this is the main engine of the Data Pump utilities. It contains procedures that do the export and import actions.
- **The DBMS_METADATA package**: this package is used to extract and modify data dictionary metadata.
- **The command-line clients, expdp and impdp.**

Data-Access Methods

- **Direct path**: the direct path internal stream format is the same format as the data stored in Oracle dump files.
- **External tables**: Oracle reads data from and write data to operating system files that lie outside the database.

Oracle always tries to first use the direct-path method. Under some conditions, such as the following, it may not be able to use the direct method:

- Clustered tables
- Presence of active triggers in the tables
- Export of a single partition in a table with a global index
- Presence of referential integrity constraints

- Presence of domain indexes on LOB columns
- Tables with fine-grained access control enabled in the insert mode
- Tables with BFILE or opaque type columns

Note: The datafile format is identical in external tables and the direct-access methods.

Data Pump Files

- **Dump files**: These hold the data for the Data Pump job.
- **Log files**: These are the standard files for logging the results of Data Pump operations.
- **SQL files**: Data Pump import uses a special parameter called `SQLFILE`, which will write all the Data Definition Language (DDL) statements it will execute during the import job to a file.

Using Directory Objects

You can't use absolute directory path location for Data Pump jobs; you must always use a directory object.

To create a directory, a user must have the `CREATE ANY DIRECTORY` privilege:

```
CREATE DIRECTORY dpump_dir1 as
'c:\oracle\product\10.1.0\oradata\export'
```

In order for a user to use a specific directory, the user must have access privileges to the directory object:

```
GRANT READ, WRITE ON DIRECTORY dpump_dir1 TO
salapati
```

1. Using the DIRECTORY:FILE Notation:

```
expdp LOGFILE=dpump_dir2:salapati.log ...
```

2. Using the DIRECTORY parameter

You can use the `DIRECTORY` parameter to specify the name of the directory object:

```
expdp hr/hr DIRECTORY=dpump_dir1 ...
```

3. Using the default directory DATA_PUMP_DIR

You can create a default directory with the name `DATA_PUMP_DIR`, and then not need to specify the `DIRECTORY` parameter in your export and import commands. Data Pump will write all dump files, SQL files, and log files automatically to the directory specified for `DATA_DUMP_DIR`.

4. Using the DATA_DUMP_DIR Environment Variable

You can use the `DATA_DUMP_DIR` environment variable on the client to point to the directory object on the server. Data Pump will automatically read and/or write its files from that directory object. In Windows, this variable is set in the Registry.

Order of Precedence for File Locations

As in the order indicated above.

The Mechanics of a Data Pump Job

The Master Process

The master process, or more accurately, the Master Control Process (MCP), has a process name of `DMnn`. The full master process name is of the format `<instance>_DMnn_<pid>`

The master process performs the following tasks:

- Creates jobs and controls them
- Creates and manages the worker processes
- Monitors the jobs and logs the progress
- Maintains the job state and restart information in the master table

- o Manages the necessary files, including the dump file set

Oracle creates the *master table* in the schema of the user who is running the Data Pump job at the beginning of every export job. The master table has the same name as the export job, such as `SYS_EXPORT_SCHEMA_01`. Master table will be automatically deleted by end of a successful export or import job.

Note: The master table contains all the necessary information to restart a stopped job. It is thus the key to Data Pump's job restart capability, whether the job stoppage is planned or unplanned.

The Worker Process

The worker process is the process that actually performs the heavy-duty work of loading and unloading data, and has the name `DWnn` (`<instance>_DWnn_<pid>`).

MCP(DMnn) may create number of `DWnn`, if you choose the `PARALLEL` option for load. `DWnn` process maintains the object rows of the master table.

Shadow Process

The shadow process creates the job consisting of the master table as well as the master process.

Client Processes

The client processes call the Data Pump's API. You perform export and import with the two clients, `expdp` and `impdp`.

Using Data Pump Export and Import

Data Pump Export Types

Using the Command Line

```
expdp system/manager directory=dpump_dir1
dumpfile=expdat1.dmp
```

Using a Parameter File

```
expdp parfile=myfile.txt
```

Using Interactive Data Pump Export

In Data Pump export, you use the interactive method for one purpose only: when you decide you need to change some export parameters midstream, while the job is still running. Note that the export or import job keeps running throughout, without any interruption.

Using EM Database Control

Start the Database Control and go to the Maintenance | Utilities page.

Data Pump Export Modes

- o **Full export** mode: using `FULL` parameter
- o **Schema** mode: using `SCHEMAS` parameter
- o **Tablespace** mode: using `TABLESPACES` and/or `TRANSPORT_TABLESPACES` parameters
- o **Table** mode: using `TABLES` parameter

Data Pump Export Parameters

File- and Directory-Related Parameters

`DIRECTORY`

specifies the location of the dump and other files.

`DUMPFIL`

provides the name of the dump file to which the export dump should be written.

You can provide multiple dump filenames in several ways:

- o by specifying the `%U` substitution variable. Using this method, the number of files you can create is equal to the value of the `PARALLEL` parameter.
- o using a comma-separated list.
- o specifying the `DUMPFIL` parameter multiple times

`FILESIZE`

this optional parameter specifies size of export file. The export job will stop if your dump file reaches its size limit.

`PARFILE`

used to specify the parameter file. Every parameter should be in a line.

`LOGFILE` and `NOLOGFILE`

You can use the `LOGFILE` parameter to specify a log file for your export jobs. If you don't specify this parameter, Oracle will create a log file named `export.log`. If you specify the parameter `NOLOGFILE`, Oracle will not create its log file.

Export Mode-Related Parameters

The export mode-related parameters are the `FULL`, `SCHEMAS`, `TABLES`, `TABLESPACES`, `TRANSPORT_TABLESPACES`, and `TRANSPORT_FULL_CHECK` parameters. The `TRANSPORT_FULL_CHECK` parameter simply checks to make sure that the tablespaces you are trying to transport meet all the conditions to qualify for the job.

Export Filtering Parameters

`CONTENT`

It controls contents of exported data. The possible values are:

- o `ALL` exports data and definitions (metadata).
- o `DATA_ONLY` exports only table rows.
- o `METADATA_ONLY` exports only metadata (this is equivalent to `rows=n`).

`EXCLUDE` and `INCLUDE`

Those are mutually exclusive parameters. The `EXCLUDE` parameter is used to omit specific database object types from an export or import operation. The `INCLUDE` parameter enables you to include only a specific set of objects.

The syntaxes of using them are as follows:

```
EXCLUDE=object_type[:name_clause]
INCLUDE=object_type[:name_clause]
```

Examples:

```
EXCLUDE=INDEX
EXCLUDE=TABLE:"LIKE 'EMP%'"
EXCLUDE=SCHEMA:"='HR'"
INCLUDE=TABLE:"IN ('EMP', 'DEPT)'"
```

`QUERY`

This parameter lets you selectively export table row data with the help of a SQL statement.

```
QUERY=OE.ORDERS: "WHERE order_id > 100000"
```

Estimation Parameters

`ESTIMATE`

The `ESTIMATE` parameter will tell you how much space your new export job is going to consume.

By default, Oracle will use the blocks method to do its estimation.

Total estimation using `BLOCKS` method: 654 KB

When you set `ESTIMATE=statistics`, Oracle will use the statistics of the database objects to calculate its estimation.

```
Total estimation using STATISTICS method:
65.72 KB
```

ESTIMATE_ONLY

Use this parameter to estimate the required export file size without starting an actual export job.

The Network Link Parameter

NETWORK_LINK

You can initiate an export job from your server and have Data Pump export data from a remote database to dump files located on the instance from which you initiate the Data Pump export job.

```
expdp hr/hr DIRECTORY=dpump_dir1
NETWORK_LINK=source_database_link
DUMPFILE=network_export.dmp
```

Interactive Mode Export Parameters

You can enter the interactive mode of Data Pump export in either of two ways:

- To get into the interactive mode, press `Ctrl+C` while the job is running.
- You can also enter the interactive mode of operation by using the `ATTACH` command.

```
expdp salapati/sammyy1
attach=SALAPATI.SYS_EXPORT_SCHEMA_01
```

You must be a DBA, or must have `EXP_FULL_DATABASE` or `IMP_FULL_DATABASE` roles, in order to attach and control Data Pump jobs of other users.

CONTINUE_CLIENT (interactive parameter)

This parameter will take you out of the interactive mode. Your client connection will still be intact, and you'll continue to see the export messages on your screen.

EXIT_CLIENT (interactive parameter)

This parameter will stop the interactive session, as well as terminate the client session.

STOP_JOB (interactive parameter)

This parameter stops running Data Pump jobs.

START_JOB (interactive parameter)

This parameter resumes stopped jobs. You can restart any job that is stopped, whether it's stopped because you issued a `STOP_JOB` command or due to a system crash, as long as you have access to the master table and an uncorrupted dump file set.

KILL_JOB (interactive parameter)

This parameter kills both the client and the Data Pump. If a job is killed using the `KILL_JOB` interactive command, the master table is dropped and the job cannot be restarted.

ADD_FILE (interactive parameter)

Use this parameter to add a dump file to your job.

```
expdp> ADD_FILE=hr2.dmp, dpump_dir2:hr3.dmp
```

HELP (can be used in interactive mode)

Displays online help.

STATUS (can be used in interactive mode)

This parameter displays detailed status of the job, along with a description of the current operation. An estimated completion percentage for the job is also returned.

In logging mode, you can assign an integer value (n) to this parameter. In this case, job status is displayed on screen every n second.

JOBNAME

Use this parameter to provide your own job name for a given Data Pump export/import job. If not provided, Oracle will give it a name of the format

```
<USER>_<OPERATION>_<MODE>_%N.
```

Example: `SYSTEM_EXPORT_FULL_01`

PARALLEL

This parameter lets you specify more than a single active execution thread for your export job. You should specify number of dump files equal to the `PARALLEL` value.

Data Pump Import Parameters

You'll need the `IMPORT_FULL_DATABASE` role to perform an import if the dump file for the import was created using the `EXPORT_FULL_DATABASE` role.

File- and Directory-Related Parameters

The Data Pump import utility uses the `PARFILE`, `DIRECTORY`, `DUMPFILE`, `LOGFILE`, and `NOLOGFILE` commands in the same way as the Data Pump export utility.

SQLFILE

This parameter enables you to extract the DDL from the export dump file, without importing any data.

```
impdp salapati/sammyy1 DIRECTORY=dpump_dir1
DUMPFILE=finance.dmp
SQLFILE=dpump_dir2:finance.sql
```

REUSE_DATAFILES

This parameter tells Data Pump whether it should use existing datafiles for creating tablespaces during an import.

Import Mode-Related Parameters

You can perform a Data Pump import in various modes, using the `TABLE`, `SCHEMAS`, `TABLESPACES`, and `FULL` parameters, just as in the case of the Data Pump export utility.

Filtering Parameters

The Data Pump import utility uses the `CONTENT`, `EXCLUDE` and `INCLUDE` parameters in the same way as the Data Pump export utility. If you use the `CONTENT=DATA_ONLY` option, you cannot use either the `EXCLUDE` or `INCLUDE` parameter during an import.

`QUERY` can also be used but in this case Data Pump will use only the external table data method, rather than the direct-path method, to access the data.

TABLE_EXISTS_ACTION

Use this parameter to tell Data Pump what to do when a table already exists.

- `SKIP` (the default), Data Pump will skip a table if it exists.
- `APPEND` value appends rows to the table.
- `TRUNCATE` value truncates the table and reloads the data from the export dump file.
- `REPLACE` value drops the table if it exists, re-creates, and reloads it.

Job-Related Parameters

The `JOB_NAME`, `STATUS`, and `PARALLEL` parameters carry identical meanings as their Data Pump export counterparts.

Import Mode-Related Parameters

You can perform a Data Pump import in various modes, using the TABLES, SCHEMAS, TABLESPACES, and FULL parameters, just as in the case of the Data Pump export utility.

Remapping Parameters

REMAP_SCHEMA

Using this parameter, you can move objects from one schema to another.

```
impdp system/manager dumpfile=newdump.dmp
REMAP_SCHEMA=hr:oe
```

REMAP_DATAFILE

Changes the name of the source datafile to the target datafile name in all SQL statements where the source datafile is referenced: CREATE TABLESPACE, CREATE LIBRARY, and CREATE DIRECTORY.

Remapping datafiles is useful when you move databases between platforms that have different file naming conventions.

```
impdp hr/hr FULL=y DIRECTORY=dpump_dir1
DUMPFILE=db_full.dmp
REMAP_DATAFILE='DB1$: [HRDATA.PAYROLL]tbs6.f':
/db1/hrdata/payroll/tbs6.f'
```

REMAP_TABLESPACE

This parameter enables you to move objects from one tablespace into a different tablespace during an import.

```
impdp hr/hr
REMAP_TABLESPACE='example_tbs': 'new_tbs'
DIRECTORY=dpump_dir1 PARALLEL=2
JOB_NAME=cfln02 DUMPFILE=employees.dmp
NOLOGFILE=Y
```

The Network Link Parameter

NETWORK_LINK

In case of network import, the server contacts the remote source database referenced by the parameter value, retrieves the data, and writes it directly back to the target database. There are no dump files involved.

```
impdp hr/hr TABLES=employees
DIRECTORY=dpump_dir1
NETWORK_LINK=source_database_link
EXCLUDE=CONSTRAINT
```

The log file is written to dpump_dir1, specified on the DIRECTORY parameter.

The TRANSFORM Parameter

TRANSFORM

This parameter instructs the Data Pump import job to modify the storage attributes of the DDL that creates the objects during the import job.

TRANSFORM = transform_name:value[:object_type]

transform_name: takes one of the following values:

SEGMENT_ATTRIBUTES

If the value is specified as *y*, then segment attributes (physical attributes, storage attributes, tablespaces, and logging) are included, with appropriate DDL. The default is *y*.

STORAGE

If the value is specified as *y*, the storage clauses are included, with appropriate DDL. The default is *y*.

This parameter is ignored if

SEGMENT_ATTRIBUTES=*n*.

object_type: The object type must be either TABLE or INDEX.

Monitoring a Data Pump Job

Viewing Data Pump Jobs

The DBA_DATAPUMP_JOBS view shows summary information of all currently running Data Pump jobs.

OWNER_NAME : User that initiated the job

JOB_NAME : Name of the job

OPERATION : Type of operation being performed

JOB_MODE : FULL, TABLE, SCHEMA, or TABLESPACE

STATE : UNDEFINED, DEFINING, EXECUTING, and NOT RUNNING.

DEGREE : Number of worker processes performing the operation

ATTACHED_SESSIONS : Number of sessions attached to the job.

Viewing Data Pump Sessions

The DBA_DATAPUMP_SESSIONS view identifies the user sessions currently attached to a Data Pump export or import job.

JOB_NAME : Name of the job

SADDR : Address of the session attached to the job.

Viewing Data Pump Job Progress

Use V\$SESSION_LONGOPS to monitor the progress of an export/import job.

TOTALWORK : shows the total estimated number of megabytes in the job.

sofar : megabytes transferred thus far in the job.

units : stands for megabytes.

opname : shows the Data Pump job name.

Creating External Tables for Data Population

Features of External Table Population Operations

- You can use the ORACLE_LOADER or ORACLE_DATAPUMP access drivers to perform data loads. You can use only the new ORACLE_DATA_PUMP access driver for unloading data (populating external tables).
- No DML or indexes are possible for external tables.
- You can use the datafiles created for an external table in the same database or a different database.

Creating External Tables

```
CREATE OR REPLACE DIRECTORY employee_data AS
'C:\employee_data'
```

```
CREATE TABLE employee_ext
(empid NUMBER(8),
 emp_name VARCHAR2(30),
 dept_name VARCHAR2(20),
 hire_date date)
ORGANIZATION EXTERNAL
(TYPE ORACLE_LOADER -- or ORACLE_DATAPUMP
DEFAULT DIRECTORY employee_data
ACCESS PARAMETERS
( RECORDS DELIMITED BY NEWLINE
FIELDS TERMINATED BY ','
MISSING FIELD VALUES ARE NULL)
LOCATION ('emp.dat'))
)
REJECT LIMIT UNLIMITED
```

Loading and Unloading Data

To load an Oracle table from an external table, you use the INSERT INTO ...SELECT clause.

To populate an external table (data unloading), you use the `CREATE TABLE AS SELECT` clause.

```
CREATE TABLE dept_xt
ORGANIZATION EXTERNAL
(
  TYPE ORACLE_DATAPUMP
  DEFAULT DIRECTORY ext_tab_dir1
  LOCATION ('dept_xt.dmp')
)
AS SELECT * FROM scott.DEPT
```

Note: If you wish to extract the metadata for any object, just use `DBMS_METADATA`, as shown here:

```
SET LONG 2000
SELECT
DBMS_METADATA.GET_DDL('TABLE', 'EXTRACT_CUST')
FROM DUAL
```

Parallel Population of External Tables

You can load external tables in a parallel fashion, simply by using the keyword `PARALLEL` when creating the external table.

The actual degree of parallelism is constrained by the number of dump files you specify under the `LOCATION` parameter.

```
CREATE TABLE inventories_xt
ORGANIZATION EXTERNAL
(
  TYPE ORACLE_DATA PUMP
  DEFAULT DIRECTORY def_dir1
  LOCATION ('inv.dmp1','inv.dmp2','inv.dmp3')
)
PARALLEL
AS SELECT * FROM inventories
```

Defining External Table Properties

The data dictionary view `DBA_EXTERNAL_TABLES` describes features of all the external tables.

```
TABLE_NAME
TYPE_OWNER
  Owner of the implementation type for the external
  table access driver
TYPE_NAME
  Name of the implementation type for the external
  table access driver
DEFAULT_DIRECTORY_OWNER
DEFAULT_DIRECTORY_NAME
REJECT_LIMIT
  Reject limit for the external table
ACCESS_TYPE
  Type of access parameters for the external table:
  BLOB or CLOB
ACCESS_PARAMETERS
  Access parameters for the external table
PROPERTY
  Property of the external table:
  o REFERENCED - Referenced columns
  o ALL (default)- All columns
```

If the `PROPERTY` column shows the value `REFERENCED`, this means that only those columns referenced by a SQL statement are processed (parsed and converted) by the Oracle access driver. `ALL` (the default) means that all the columns will be processed even those not existing in the select list.

To change the `PROPERTY` value for a table:

```
ALTER TABLE dept_xt
PROJECT COLUMN REFERENCED
```

Transporting Tablespaces Across Platforms

Introduction to Transportable Tablespaces

In Oracle Database 10g, you can transport tablespaces between different platforms.

Transportable tablespaces are a good way to migrate a database between different platforms.

You must be using the Enterprise Edition of Oracle8i or higher to generate a transportable tablespace set. However, you can use any edition of Oracle8i or higher to plug a transportable tablespace set into an Oracle Database on the same platform.

To plug a transportable tablespace set into an Oracle Database on a different platform, both databases must have compatibility set to at least 10.0.

Many, but not all, platforms are supported for cross-platform tablespace transport. You can query the `V$TRANSPORTABLE_PLATFORM` view to see the platforms that are supported.

Limitations on Transportable Tablespace Use

- The source and target database must use the same character set and national character set.
- Objects with underlying objects (such as materialized views) or contained objects (such as partitioned tables) are not transportable unless all of the underlying or contained objects are in the tablespace set.
- You cannot transport the `SYSTEM` tablespace or objects owned by the user `SYS`.

Transporting Tablespaces Between Databases

1. Check *endian* format of both platforms. For cross-platform transport, check the endian format of both platforms by querying the `V$TRANSPORTABLE_PLATFORM` view. You can find out your own platform name:

```
select platform_name from v$database
```

2. Pick a self-contained set of tablespaces.

The following statement can be used to determine whether tablespaces `sales_1` and `sales_2` are self-contained, with referential integrity constraints taken into consideration:

```
DBMS_TTS.TRANSPORT_SET_CHECK( TS_LIST
=>'sales_1,sales_2', INCL_CONSTRAINTS =>TRUE,
FULL_CHECK =>TRUE)
```

Note: You must have been granted the `EXECUTE_CATALOG_ROLE` role (initially signed to `SYS`) to execute this procedure.

You can see all violations by selecting from the `TRANSPORT_SET_VIOLATIONS` view. If the set of tablespaces is self-contained, this view is empty.

3. Generate a transportable tablespace set.
 - 3.1. Make all tablespaces in the set you are copying read-only.
 - 3.2. Export the metadata describing the objects in the tablespace(s)

```
EXPDP system/password
DUMPFIL=expdat.dmp DIRECTORY=dpump_dir
TRANSPORT_TABLESPACES = sales_1,sales_2
TRANSPORT_FULL_CHECK=Y
```

3.3. If you want to convert the tablespaces in the source database, use the RMAN

```
RMAN TARGET /
CONVERT TABLESPACE sales_1,sales_2
TO PLATFORM 'Microsoft Windows NT'
FORMAT '/temp/%U'
```

4. Transport the tablespace set.

Transport both the datafiles and the export file of the tablespaces to a place accessible to the target database.

5. Convert tablespace set, if required, in the destination database.

Use RMAN as follows:

```
RMAN> CONVERT DATAFILE
'/hq/finance/work/tru/tbs_31.f',
'/hq/finance/work/tru/tbs_32.f',
'/hq/finance/work/tru/tbs_41.f'
TO PLATFORM="Solaris[tm] OE (32-bit)"
FROM PLATFORM="HP TRu64 UNIX"
DBFILE_NAME_CONVERT=
"/hq/finance/work/tru/",
"/hq/finance/dbs/tru"
PARALLELISM=5
```

Note: The source and destination platforms are optional.

Note: By default, Oracle places the converted files in the Flash Recovery Area, without changing the datafile names.

6. Plug in the tablespace.

```
IMPDP system/password DUMPFILE=expdat.dmp
DIRECTORY=dpump_dir
TRANSPORT_DATAFILES=
/salesdb/sales_101.dbf,
/salesdb/sales_201.dbf
REMAP_SCHEMA=(dcranney:smith)
REMAP_SCHEMA=(jfee:williams)
```

If required, put the tablespace into READ WRITE mode.

A Few Restrictions

- There are a few restrictions on what tablespaces can qualify for transportability:
- You cannot transport the SYSTEM tablespace or any of its contents. This means that you cannot use TTS for PL/SQL, triggers, or views. These would have to be moved with export.
- The source and target database must have the same character set and national language set.
- You cannot transport a table with a materialized view unless the mview is in the transport set you create.
- You cannot transport a partition of a table without transporting the entire table.

Using Transportable Tablespaces: Scenarios

Transporting and Attaching Partitions for Data Warehousing

1. In a staging database, you create a new tablespace and make it contain the table you want to transport. It should have the same columns as the destination partitioned table.
2. Create an index on the same columns as the local index in the partitioned table.
3. Transport the tablespace to the data warehouse.
4. In the data warehouse, add a partition to the table.

```
ALTER TABLE sales ADD PARTITION jul98 VALUES
LESS THAN (1998, 8, 1)
```

5. Attach the transported table to the partitioned table by exchanging it with the new partition:

```
ALTER TABLE sales EXCHANGE PARTITION jul98
WITH TABLE jul_sales
INCLUDING INDEXES WITHOUT VALIDATION
```

Publishing Structured Data on CDs

A data provider can load a tablespace with data to be published, generate the transportable set, and copy the transportable set to a CD. When customers receive this CD, they can plug it into an existing database without having to copy the datafiles from the CD to disk storage.

Note: In this case, it is highly recommended to set the READ_ONLY_OPEN_DELAYED initialization parameter to TRUE.

Mounting the Same Tablespace Read-Only on Multiple Databases

You can use transportable tablespaces to mount a tablespace read-only on multiple databases.

Moving Databases Across Platforms Using Transportable Tablespaces

You cannot transport the SYSTEM tablespace.

Therefore, objects such as sequences, PL/SQL packages, and other objects that depend on the SYSTEM tablespace are not transported.

Archiving Historical Data Using Transportable Tablespaces

Using Transportable Tablespaces to Perform TSPITR

Automatic Database Management

Using the Automatic Database Diagnostic Monitor (ADDM)

The *Automatic Workload Repository (AWR)* is a statistics collection facility that collects new performance statistics in the form of a snapshot on an hourly basis and saves the snapshots for seven days into SYSAUX before purging them.

The *Automatic Database Diagnostic Monitor (ADDM)* is a new diagnosis tool that runs automatically every hour, after the AWR takes a new snapshot. The ADDM uses the AWR performance snapshots to locate the root causes for poor performance and saves recommendations for improving performance in SYSAUX.

You can then go to the OEM Database Control to view the results, or even view them from a SQL*Plus session with the help of an Oracle-supplied SQL script.

Goal of the ADDM

ADD aims at reducing a key database metric called *DB time*, which stands for the cumulative amount of time (in milliseconds) spent on actual database calls (at the user level); i.e. both the wait time and processing time (CPU time).

Problems That the ADDM Diagnoses

- Configuration issues
- Improper application usage
- Expensive SQL statements
- I/O performance issues
- Locking issues
- Excessive parsing

- CPU bottlenecks
- Undersized memory allocation
- Connection management issues, such as excessive logon/logoff statistics

The New Time Model

V\$SYS_TIME_MODEL

This view shows time in terms of the number of microseconds the database has spent on a specific operation.

V\$SESS_TIME_MODEL

displays the same information in the session-level.

Automatic Management of the ADDM

The Manageability Monitor Process (MMON) process schedules the automatic running of the ADDM.

Configuring the ADDM

You only need to make sure that the initialization parameters `STATISTICS_LEVEL` is set to `TYPICAL` or `ALL`, in order for the AWR to gather its cache of performance statistics.

Determining Optimal I/O Performance

Oracle assumes the value of the parameter (not initialization parameter) `DBIO_EXPECTED` is 10 milliseconds.

```
SELECT PARAMETER_VALUE
FROM DBA_ADVISOR_DEF_PARAMETERS
WHERE ADVISOR_NAME='ADDM'
AND PARAMETER_NAME='DBIO_EXPECTED'
```

If your hardware is significantly different, you can set the parameter value one time for all subsequent ADDM executions:

```
DBMS_ADVISOR.SET_DEFAULT_TASK_PARAMETER('ADDM',
'DBIO_EXPECTED', 8000);
```

Running the ADDM

MMON schedules the ADDM to run every time the AWR collects its most recent snapshot.

To view the ADDM's findings:

- Use the OEM Database Control
- Run the Oracle-provided script `addmrpt.sql`

The ADDM Analysis

ADDM analysis finding consists of the following four components:

- The definition of the problem itself
- The root cause of the performance problem
- Recommendation(s) to fix the problem
- The rationale for the proposed recommendations

Viewing Detailed ADDM Reports

Click the View Report button on the ADDM main page in the Database Control.

Using the DBMS_ADVISOR Package to Manage the ADDM

The `DBMS_ADVISOR` package is part of the Server Manageability Suite of advisors, which is a set of rule-based expert systems that identify and resolve performance problems of several database components.

Note: The `DBMS_ADVISOR` package requires the `ADVISOR` privilege.

```
CREATE_TASK          to create a new advisor task.
SET_DEFAULT_TASK    helps you modify default values of
                    parameters within a task.
```

```
DELETE_TASK         deletes a specific task from the
                    repository.
EXECUTE_TASK        executes a specific task.
GET_TASK_REPORT     displays the most recent ADDM
                    report.
SET_DEFAULT_TASK_PARAMETER modifies a default task parameter.
```

Syntaxes:

```
DBMS_ADVISOR.GET_TASK_REPORT (
task_name ,
type , -- TEXT, XML, HTML
level, -- TYPICAL, ALL, BASIC
section, owner_name) RETURN CLOB
```

Examples:

```
CREATE OR REPLACE FUNCTION run_addm(start_time
IN DATE, end_time IN DATE )
RETURN VARCHAR2
IS
begin_snap NUMBER;
end_snap NUMBER;
tid NUMBER; -- Task ID
tname VARCHAR2(30); -- Task Name
tdesc VARCHAR2(256); -- Task Description
BEGIN
-- Find the snapshot IDs corresponding to the
-- given input parameters.
SELECT max(snap_id) INTO begin_snap
FROM DBA_HIST_SNAPSHOT
WHERE trunc(end_interval_time, 'MI') <=
start_time;
SELECT min(snap_id) INTO end_snap
FROM DBA_HIST_SNAPSHOT
WHERE end_interval_time >= end_time;
--
-- set Task Name (tname) to NULL and let
-- create_task return a unique name for
-- the task.
tname := '';
tdesc := 'run_addm( ' || begin_snap || ', ' ||
end_snap || ' )';
--
-- Create a task, set task parameters and
-- execute it
DBMS_ADVISOR.CREATE_TASK( 'ADDM', tid, tname,
tdesc );
DBMS_ADVISOR.SET_TASK_PARAMETER( tname,
'START_SNAPSHOT', begin_snap );
DBMS_ADVISOR.SET_TASK_PARAMETER( tname,
'END_SNAPSHOT' , end_snap );
DBMS_ADVISOR.EXECUTE_TASK( tname );
RETURN tname;
END;
/

SET PAGESIZE 0 LONG 1000000 LONGCHUNKSIZE 1000
COLUMN get_clob FORMAT a80

-- execute run_addm() with 7pm and 9pm as
-- input
VARIABLE task_name VARCHAR2(30);
BEGIN
:task_name := run_addm( TO_DATE('19:00:00
(10/20)', 'HH24:MI:SS (MM/DD)'),
TO_DATE('21:00:00 (10/20)', 'HH24:MI:SS
(MM/DD)' ) );
END;
/
-- execute GET_TASK_REPORT to get the textual
-- ADDM report.
SELECT
DBMS_ADVISOR.GET_TASK_REPORT(:task_name)
```

```
FROM DBA_ADVISOR_TASKS t
WHERE t.task_name = :task_name
AND t.owner = SYS_CONTEXT( 'userenv',
'session_user' );
```

ADDM-Related Dictionary Views

```
DBA_ADVISOR_RECOMMENDATIONS
DBA_ADVISOR_FINDINGS
DBA_ADVISOR_RATIONALE
```

Using Automatic Shared Memory Management

With Automatic Shared Memory Management, Oracle will use internal views and statistics to decide on the best way to allocate memory among the SGA components. The new process MMAN constantly monitors the workload of the database and adjusts the size of the individual memory components accordingly.

Note: In Oracle Database 10g, the database enables the Automatic PGA Memory Management feature by default. However, if you set the `PGA_AGGREGATE_TARGET` parameter to 0 or the `WORKAREA_SIZE_POLICY` parameter to `MANUAL`, Oracle doesn't use Automatic PGA Memory Management.

Manual Shared Memory Management

As in previous version, you use the following parameters to set SGA component sizes:

```
DB_CACHE_SIZE, SHARED_POOL_SIZE, LARGE_POOL,
JAVA_POOL_SIZE, LOG_BUFFER and
STREAMS_POOL_SIZE.
```

In Oracle Database 10g, the value of the `SHARED_POOL_SIZE` parameter includes the internal overhead allocations for metadata such as the various data structures for sessions and processes.

You must, therefore, make sure to increase the size of the `SHARED_POOL_SIZE` parameter when you are upgrading to Oracle Database 10g. You can find the appropriate value by using the following query:

```
select sum(BYTES)/1024/1024 from V$SGASTAT
where POOL = 'shared pool'
```

Automatic Memory Management

`SGA_TARGET` specifies the total size of all SGA components. If `SGA_TARGET` is specified, then the following memory pools are automatically sized:

- o Buffer cache (`DB_CACHE_SIZE`)
- o Shared pool (`SHARED_POOL_SIZE`)
- o Large pool (`LARGE_POOL_SIZE`)
- o Java pool (`JAVA_POOL_SIZE`)

If these automatically tuned memory pools are set to non-zero values, then those values are used as minimum levels by Automatic Shared Memory Management.

The following pools are not affected by Automatic Shared Memory Management:

- o Log buffer
- o Other buffer caches, such as `KEEP`, `RECYCLE`, and other block sizes
- o Streams pool
- o Fixed SGA and other internal allocations
- o The new Oracle Storage Management (OSM) buffer cache, which is meant for the optional ASM instance

The memory allocated to these pools is deducted from the total available for `SGA_TARGET` when Automatic

Shared Memory Management computes the values of the automatically tuned memory pools.

Note: If you dynamically set `SGA_TARGET` to zero, the size of the four auto-tuned shared memory components will remain at their present levels.

Note: The `SGA_MAX_SIZE` parameter sets an upper bound on the value of the `SGA_TARGET` parameter.

Note: In order to use Automatic Shared Memory Management, you should make sure that the initialization parameter `STATISTICS_LEVEL` is set to `TYPICAL` or `ALL`.

You can use the `V$SGA_DYNAMIC_COMPONENTS` view to see the values assigned to the auto-tuned components.

When you restart the instance, by using `SPFILE` Oracle will start with the values the auto-tuned memory parameters had before you shut down the instance.

Using Automatic Optimizer Statistics Collection

All you need to do to make sure the automatic statistics collection process works is to ensure that the `STATISTICS_LEVEL` initialization parameter is set to `TYPICAL` or `ALL`.

Oracle will use the `DBMS_STATS` package to collect optimizer statistics on an automatic basis.

Using the Scheduler to Run DBMS_GATHER_STATS_JOB

Oracle automatically creates a database job called `GATHER_STATS_JOB` at database creation time.

```
select JOB_NAME
from DBA_SCHEDULER_JOBS
where JOB_NAME like 'GATHER_STATS%'
```

Oracle automatically schedules the `GATHER_STATS_JOB` job to run when the maintenance window opens.

The `GATHER_STATS_JOB` job calls the procedure `DBMS_STATS.GATHER_DATABASE_STATS_JOB_PROC` to gather the optimizer statistics.

The job collects statistics only for objects with missing statistics and objects with stale statistics.

If you want to stop the automatic gathering of statistics:

```
DBMS_SCHEDULER.DISABLE('GATHER_STATS_JOB')
```

Using the Database Control to Manage the GATHER_STATS_JOB Schedule

1. click the **Administration tab**.
2. **Scheduler Group -> Windows Link**
3. Click the **Edit** button. You'll then be able to edit the weeknight or the weekend window timings.

Table Monitoring

You cannot use the `ALTER_DATABASE_TAB_MONITORING` and `ALTER_SCHEMA_TAB_MONITORING` procedures of the `DBMS_STATS` package to turn table monitoring on and off at the database and schema level, respectively, because these subprograms are deprecated in Oracle Database 10g. Oracle 10g automatically performs these functions.

Manual Collection of Optimizer Statistics

Oracle 10g allows you to gather Optimizer statistics manually using the `DBMS_STATS`.

Handling Volatile Tables by Locking Statistics

You can lock statistics of specific objects so that current object statistics will be used by the optimizer regardless of data changes on the locked objects.

Use the following procedures in `DBMS_STATS`

- o `LOCK_TABLE_STATISTICS`
- o `UNLOCK_TABLE_STATISTICS`
- o `LOCK_SCHEMA_STATISTICS`
- o `UNLOCK_SCHEMA_STATISTICS`

Example:

```
DBMS_STATS.LOCK_TABLE_STATS('scott','test')
```

Overriding Statistics Locking

You may want Oracle to override any existing statistics locks. You can do so by setting the `FORCE` argument with several procedures to `TRUE` in the `DBMS_STATS` package. The default is `FALSE`.

Restoring Historical Optimizer Statistics

Fortunately, Oracle lets you automatically save all old statistics whenever you refresh the statistics.

You can restore statistics by using the appropriate `RESTORE_*_STATS` procedures.

The view `DBA_OPTSTAT_OPERATIONS` contains a history of all optimizer statistics collections.

```
DBA_TAB_STATS_HISTORY
```

This view contains a record of all changes made to table statistics. By default, the `DBA_TAB_STATS_HISTORY` view saves the statistics history for 31 days. However, by using the `ALTER_STATS_HISTORY_RETENTION` procedure of the `DBMS_STATS` package, you can change the default value of the statistics history retention interval.

Using Automatic Undo Retention Tuning

Oracle recommends using Automatic Undo Management (AUM) feature. However, be aware that the Manual undo management is the default.

AUM is controlled by the following parameters:

- o `UNDO_MANAGEMENT` : `AUTO|MANUAL`
- o `UNDO_TABLESPACE`
- o `UNDO_RETENTION` : default is 900 seconds

The Undo Advisor

This OEM utility provides you undo related functions like:

- o undo tablespace size recommendations
- o undo retention period recommendations

Using the Retention Guarantee Option

This feature guarantees that Oracle will never overwrite any undo data that is within the undo retention period. This new feature is disabled by default. You can enable the guarantee feature at database creation time, at the undo tablespace creation time, or by using the alter tablespace command.

```
alter tablespace UNDOTBS1 retention guarantee
```

Manageability Infrastructure

Types of Oracle Statistics

Cumulative Statistics

Cumulative statistics are the accumulated total value of a particular statistic since instance startup.

Database Metrics

Database metrics are the statistics that measure the rate of change in a cumulative performance statistic.

The background process MMON (Manageability Monitor) updates metric data on a minute-by-minute basis, after collecting the necessary fresh base statistics.

Sample Data

The new Automatic Session History (ASH) feature now automatically collects session sample data, which represents a sample of the current state of the active sessions.

Baseline Data

The statistics from the period where the database performed well are called baseline data.

MMON process takes snapshots of statistics and save them into disks.

The Manageability Monitor Light (MMNL) process performs:

- o computing metrics
- o capturing session history information for the Automatic Session History (ASH) feature under some circumstances. For example, the MMNL process will flush ASH data to disk if the ASH memory buffer fills up before the one hour interval that would normally cause MMON to flush it.

The Automatic Workload Repository (AWR)

Its task is the automatic collection of performance statistics in the database.

AWR provides performance statistics in two distinct formats:

- A temporary in-memory collection of statistics in the SGA, accessible by (`v$`) views.
- A persistent type of performance data in the form of regular AWR snapshots, accessible by (`DBA_*`) views.

Using the `DBMS_WORKLOAD_REPOSITORY` Package to Manage AWR Snapshots

To manually creating a snapshot:

```
dbms_workload_repository.create_snapshot()
```

To drop a range of snapshots:

```
dbms_workload_repository.drop_snapshot_range(
low_snap_id => 40,high_snap_id => 60, dbid =>
2210828132)
```

To modify a AWR setting:

```
DBMS_WORKLOAD_REPOSITORY.MODIFY_SNAPSHOT_SETTING(
retention => 43200, interval => 30, dbid =>
3310949047)
```

In this example, the retention period is specified as 43200 minutes (30 days) and the interval between each snapshot is specified as 30 minutes.

Note: If you set the value of the `RETENTION` parameter to zero, you disable the automatic purging of the AWR. If you set the value of the `INTERVAL` parameter to zero, you disable the automatic capturing of AWR snapshots.

Creating and Deleting AWR Snapshot Baselines

Whenever you create a baseline by defining it over any two snapshots (identified by their snap IDs), the AWR retains the snapshots indefinitely (it won't purge these snapshots after the default period of seven days), unless you decide to drop the baseline itself.

To create a new snapshot baseline:

```
dbms_workload_repository.create_baseline(
start_snap_id => 125, end_snap_id => 185,
```

```
baseline_name => 'peak_time baseline', dbid =>
2210828132)
```

To drop a snapshot baseline:

```
dbms_workload_repository.drop_baseline
(baseline_name => 'peak_time baseline', cascade
=> FALSE, dbid => 2210828132)
```

By setting `CASCADE` parameter to `TRUE`, you can drop the actual snapshots as well.

Note: If AWR does not find room in the `SYSAUX` tablespace, Oracle will start deleting oldest snapshot regardless of values of `INTERVAL` and `RETENTION`.

Creating AWR Reports

Use the script `awrrpt.sql` to generate summary reports about the statistics collected by the AWR facility.

Note: You must have the `SELECT ANY DICTIONARY` privilege in order to run the `awrrpt.sql` script.

AWR Statistics Data Dictionary Views

<code>DBA_HIST_SNAPSHOT</code>	shows all snapshots saved in the AWR.
<code>DBA_HIST_WR_CONTROL</code>	displays the settings to control the AWR.
<code>DBA_HIST_BASELINE</code>	shows all baselines and their beginning and ending snap ID numbers.

Active Session History (ASH)

Oracle Database 10g now collects the Active Session History (ASH) statistics (mostly the wait statistics for different events) for all active sessions every second, and stores them in a circular buffer in the SGA.

The ASH feature uses about 2MB of SGA memory per CPU.

Current Active Session Data

`V$ACTIVE_SESSION_HISTORY` enables you to access the ASH statistics.

A database session is considered active if it was on the CPU or was waiting for an event that didn't belong to the Idle wait class (indicated by `SESSION_STATE` column).

DBA_HIST_ACTIVE_SESSION_HISTORY View

This view in fact is a collection of snapshots from the `V$ACTIVE_SESSION_HISTORY` view. It is populated either by MMON during its regular snapshot capturing or by MMNL when the memory buffer is full.

Server-Generated Alerts

Introduction to Metrics

MMON collects database metrics continuously and automatically saves them in the SGA for one hour.

The OEM Database Control's All Metrics page offers an excellent way to view the various metrics.

Oracle Database 10g Metric Groups are (can be obtained from `V$METRICGROUP`):

- o Event Class Metrics
- o Event Metrics
- o File Metrics
- o Service Metrics
 - `V$SERVICEMETRIC`, `V$SERVICEMETRIC_HISTORY`
- o Session Metrics
- o System Metrics

```
V$SYSMETRIC, V$SYSMETRIC_HISTORY
```

o Tablespace Metrics

Viewing Saved Metrics

MMON will automatically flush the metric data from the SGA to the `DBA_HISTORY_*` views on disk. Examples of the history views are `DBA_HIST_SUMMARY_HISTORY`, `DBA_HIST_SYSMETRIC_HISTORY`, and `DBA_HIST_METRICNAME`. Each of these views contains snapshots of the corresponding `v$` view.

Database Alerts

There are three situations when a database can send an alert:

- A monitored metric crosses a critical threshold value
- A monitored metric crosses a warning threshold value
- A service or target suddenly becomes unavailable

Default Server-Generated Alerts

Your database comes with a set of the following default alerts already configured. In addition, you can choose to have other alerts.

- Any snapshot too old errors
- Tablespace space usage (warning alert at 85 percent usage; critical alert at 97 percent usage)
- Resumable session suspended
- Recovery session running out of free space

Server-Generated Alert Mechanism

MMON process checks all the configured metrics and if any metric crosses a preset threshold, an alert will be generated.

Using the Database Control to Manage Server Alerts

You can use Database Control to:

- set a warning and critical threshold
- A response action: a SQL script or a OS command line to execute
- set Notification Rules: when notify a DBA

Using the DBMS_SERVER_ALERT Package to Manage Alerts

```
SET_THRESHOLD
```

This procedure will set warning and critical thresholds for given metrics.

```
DBMS_SERVER_ALERT.SET_THRESHOLD(
DBMS_SERVER_ALERT.CPU_TIME_PER_CALL,
DBMS_SERVER_ALERT.OPERATOR_GE, '8000',
DBMS_SERVER_ALERT.OPERATOR_GE, '10000', 1, 2,
'inst1',
DBMS_SERVER_ALERT.OBJECT_TYPE_SERVICE,
'dev.oracle.com')
```

In this example, a warning alert is issued when CPU time exceeds 8000 microseconds for each user call and a critical alert is issued when CPU time exceeds 10,000 microseconds for each user call. The arguments include:

- o `CPU_TIME_PER_CALL` specifies the metric identifier. For a list of support metrics, see PL/SQL Packages and Types Reference.
- o The observation period is set to 1 minute. This period specifies the number of minutes that the condition must deviate from the threshold value before the alert is issued.
- o The number of consecutive occurrences is set to 2. This number specifies how many times the metric

value must violate the threshold values before the alert is generated.

- o The name of the instance is set to inst1.
- o The constant `DBMS_ALERT.OBJECT_TYPE_SERVICE` specifies the object type on which the threshold is set. In this example, the service name is `dev.oracle.com`.

Note: If you don't want Oracle to send any metric-based alerts, simply set the warning value and the critical value to `NULL`.

`GET_THRESHOLD`

Use this procedure to retrieve threshold values.

```
DBMS_SERVER_ALERT.GET_THRESHOLD(
  metrics_id IN NUMBER,
  warning_operator OUT NUMBER,
  warning_value OUT VARCHAR2,
  critical_operator OUT NUMBER,
  critical_value OUT VARCHAR2,
  observation_period OUT NUMBER,
  consecutive_occurrences OUT NUMBER,
  instance_name IN VARCHAR2,
  object_type IN NUMBER,
  object_name IN VARCHAR2)
```

Using the Alert Queue

You can use the `DBMS_AQ` and `DBMS_AQADM` packages for directly accessing and reading alert messages in the alert queue.

Steps you should follow are:

1. Create an agent and subscribe the agent to the `ALERT_QUEUE` using the `CREATE_AQ_AGENT` and `ADD_SUBSCRIBER` procedures of the `DBMS_AQADM` package.
2. Associate a database user with the subscribing agent and assign the enqueue privilege to the user using the `ENABLE_DB_ACCESS` and `GRANT_QUEUE_PRIVILEGE` procedures of the `DBMS_AQADM` package.
3. Optionally, you can register with the `DBMS_AQ.REGISTER` procedure to receive an asynchronous notification when an alert is enqueued to `ALERT_QUEUE`.
4. To read an alert message, you can use the `DBMS_AQ.DEQUEUE` procedure or `OCI AQDeq` call. After the message has been dequeued, use the `DBMS_SERVER_ALERT.EXPAND_MESSAGE` procedure to expand the text of the message.

Data Dictionary Views of Metrics and Alerts

<code>DBA_THRESHOLDS</code>	lists the threshold settings defined for the instance.
<code>DBA_OUTSTANDING_ALERTS</code>	describes the outstanding alerts in the database.
<code>DBA_ALERT_HISTORY</code>	lists a history of alerts that have been cleared.
<code>V\$ALERT_TYPES</code>	provides information such as group and type for each alert.
<code>V\$METRICNAME</code>	contains the names, identifiers, and other information about the system metrics.
<code>V\$METRIC</code> and <code>V\$METRIC_HISTORY</code>	views contain system-level metric values in memory.

`V$ALERT_TYPES`

`STATE`

Holds two possible values: `stateful` or `stateless`.

The database considers all the non-threshold alerts as stateless alerts. A stateful alert first appears in the

`DBA_OUTSTANDING_ALERTS` view and goes to the `DBA_ALERT_HISTORY` view when it is cleared. A stateless alert goes straight to `DBA_ALERT_HISTORY`.

`SCOPE`

Classifies alerts into database wide and instance wide. The only database-level alert is the one based on the *Tablespace Space Usage* metric. All the other alerts are at the instance level.

`GROUP_NAME`

Oracle aggregates the various database alerts into some common groups: Space, Performance, Configuration-related database alerts.

The Management Advisory Framework

The Advisors

Memory-Related Advisors

- Buffer Cache Advisor
- Library Cache Advisor
- PGA Advisor

Space-Related Advisors

- Segment Advisor
- Undo Advisor

Tuning-Related Advisors

- SQL Tuning Advisor
- SQL Access Advisor

Using the `DBMS_ADVISOR` Package

You can run any of the advisors using the `DBMS_ADVISOR` package.

Prerequisite: `ADVISOR` privilege.

The following are the steps you must follow:

1. Creating a Task

```
VARIABLE task_id NUMBER;
VARIABLE task_name VARCHAR2(255);
EXECUTE :task_name := 'TEST_TASK';
EXECUTE DBMS_ADVISOR.CREATE_TASK ('SQL Access Advisor', :task_id, :task_name);
```

2. Defining the Task Parameters: The task parameters control the recommendation process. The parameters you can modify belong to four groups: workload filtering, task configuration, schema attributes, and recommendation options.

```
Example: DBMS_ADVISOR.SET_TASK_PARAMETER (
  'TEST_TASK', 'VALID_TABLE_LIST', 'SH.SALES, SH.CUSTOMERS');
```

3. Generating the Recommendations

```
DBMS_ADVISOR.EXECUTE_TASK('TEST_TASK');
```

4. Viewing the Recommendations: You can view the recommendations of the advisor task by using the `GET_TASK_REPORT` procedure or querying the `DBA_ADVISOR_RECOMMENDATIONS` view.

Using the Database Control to Manage the Advisory Framework

Click the **Advisor Central** link on the Database Control home page.

Dictionary Views related to the Advisors

```
DBA_ADVISOR_TASKS
DBA_ADVISOR_PARAMETERS
DBA_ADVISOR_FINDINGS
DBA_ADVISOR_RECOMMENDATIONS
DBA_ADVISOR_ACTIONS
DBA_ADVISOR_RATIONALE
```

Application Tuning

Using the New Optimizer Statistics

- The default value for the `OPTIMIZER_MODE` initialization parameter is `ALL_ROWS`.
- Automatic Statistics Collection
- Changes in the `DBMS_STATS` Package
- Dynamic Sampling
Oracle determines at compile time whether a query would benefit from dynamic sampling.
Depending on the value of the `OPTIMIZER_DYNAMIC_SAMPLING` initialization parameter, a certain number of blocks are read by the dynamic sampling query to estimate statistics.
`OPTIMIZER_DYNAMIC_SAMPLING` takes values from zero (OFF) to 10 (default is 2).
- Table Monitoring
If you use either the `GATHER AUTO` or `STALE` settings when you use the `DBMS_STATS` package, you don't need to explicitly enable table monitoring in Oracle Database 10g; the `MONITORING` and `NO MONITORING` keywords are deprecated.
Oracle uses the `DBA_TAB_MODIFICATIONS` view to determine which objects have stale statistics.
Setting the `STATISTICS_LEVEL` to `BASIC` turns off the default table monitoring feature.
- Collection for Dictionary Objects
You can gather *fixed object* statistics by using the `GATHER_DATABASE_STATS` procedure and setting the `GATHER_FIXED` argument to `TRUE` (the default is `FALSE`).
You can also use the new procedure:
`DBMS_STATS.GATHER_FIXED_OBJECTS_STATS ('ALL')`
You must have the `SYSDBA` or `ANALYZE ANY` `DICTIONARY` system privilege to analyze any dictionary objects or fixed objects.
To collect statistics for the *real* dictionary tables:
 - Use the `DBMS_STATS.GATHER_DATABASE_STATS` procedure, by setting the `GATHER_SYS` argument to `TRUE`. Alternatively, you can use the `GATHER_SCHEMA_STATS ('SYS')` option.
 - Use the `DBMS_STATS.GATHER_DICTIONARY_STATS` procedure.

Using the SQL Tuning Advisor

Providing SQL Statements to the SQL Tuning Advisor

- Create a new set of statements as an input for the SQL Tuning Advisor.
- The ADDM may often recommend high-load statements.
- Choose a SQL statement that's stored in the AWR.
- Choose a SQL statement from the database cursor cache.

How the SQL Tuning Advisor Works

The optimizer will work in the new tuning mode wherein it conducts an in-depth analysis to come up with a set of recommendations, the rationale for them and the expected benefit if you follow the recommendations.

When working in tuning mode, the optimizer is referred to as the Automatic Tuning Optimizer (ATO).

The ATO performs the following tuning tasks:

- Statistics analysis

- SQL profiling
- Access path analysis
- SQL structure analysis

Statistics Analysis

ATO recommends collecting new statistics for specific objects, if required.

SQL Profiling

The ATO's goal at this stage is to verify that its own estimates of factors like column selectivity and cardinality of database objects are valid.

- Dynamic data sampling
Using a sample of the data, the ATO can check if its own estimates for the statement in question are significantly off the mark.
- Partial execution
The ATO may partially execute a SQL statement, so it can check if whether a plan derived purely from inspection of the estimated statistics is actually the best plan.
- Past execution history statistics
The ATO may also use any existing history of the SQL statement's execution to determine appropriate settings for parameters like `OPTIMIZER_MODE`.

The output of this phase is a *SQL Profile* of the concerned SQL statement. If you create that SQL profile, it will be used later by the optimizer when it executes the same SQL statement in the normal mode. A SQL profile is simply a set of auxiliary or supplementary information about a SQL statement.

Access Path Analysis

The ATO analyzes the potential impact of using improved access methods, such as additional or different indexes.

SQL Structure Analysis

The ATO may also make recommendations to modify the structure, both the syntax and semantics, in your SQL statements.

SQL Tuning Advisor Recommendations

The SQL Tuning Advisor can recommend that you do the following:

- Create indexes to speed up access paths
- Accept a SQL profile, so you can generate a better execution plan
- Gather optimizer statistics for objects with no or stale statistics
- Rewrite queries based on the advisor's advice

Using the SQL Tuning Advisor

Using the `DBMS_SQLTUNE` Package

The `DBMS_SQLTUNE` package is the main Oracle Database 10g interface to tune SQL statements.

Following are the required steps:

1. Create a task. You can use the `CREATE_TUNING_TASK` procedure to create a task to tune either a single statement or several statements.

```
execute :v_task :=  
DBMS_SQLTUNE.CREATE_TUNING_TASK(sql_text=>'select  
count(*) from hr.employees,hr.dept')
```

2. Execute the task. You start the tuning process by running the `EXECUTE_TUNING_TASK` procedure.

```
SET LONG 1000  
SET LONGCHUNKSIZE 1000  
SET LINESIZE 100  
SELECT DBMS_SQLTUNE.REPORT_TUNING_TASK(  
:v_task) FROM DUAL;
```

3. Get the tuning report. By using the `REPORT_TUNING_TASK` procedure.
4. Use `DROP_TUNING_TASK` to drop a task, removing all results associated with the task.

Managing SQL Profiles

Use the `DBMS_SQLTUNE.ACCEPT_SQL_PROFILE` procedure to create a SQL profile based on the recommendations of the ATO.

Managing SQL Tuning Categories

- Any created SQL Profile will be assigned to a category defined by the parameter `SQLTUNE_CATEGORY`.
- By default, `SQLTUNE_CATEGORY` has the value of `DEFAULT`.
- You can change the SQL tuning category for all users with the following command:

```
ALTER SYSTEM SET SQLTUNE_CATEGORY = PROD
```
- To change a session's tuning category, use the following command:

```
ALTER SESSION SET SQLTUNE_CATEGORY = DEV
```

You may also use the `DBMS_SQLTUNE.ALTER_SQL_PROFILE` procedure to change the SQL tuning category.

Using the Database Control to Run the SQL Tuning Advisor

Under the **Performance** tab, click the **Advisor Central** link and then click the **SQL Tuning Advisor** link.

There are several possible sources for the tuning advisor's SQL Tuning Set (STS) input:

- high-load SQL statements identified by the ADDM
- statements in the cursor cache
- statements from the AWR
- a custom workload
- another new STS.

Using the SQL Access Advisor

The SQL Access Advisor primarily provides advice regarding the creation of indexes, materialized views, and materialized view logs, in order to improve query performance.

Providing Input for the SQL Access Advisor

There are four main sources of input for the advisor: SQL cache, user-defined workload, hypothetical workload, and STS from the AWR.

Modes of Operation

You can operate the SQL Access Advisor in two modes:

Limited (partial)

In this mode, the advisor will concern itself with only problematic or high cost SQL statements ignoring statements with a cost below a certain threshold.

Comprehensive (full)

In this mode, the advisor will perform a complete and exhaustive analysis of all SQL statements in a representative set of SQL statements, after considering the impact on the entire workload.

You can also use workload filters to specify which kinds of SQL statements the SQL Access Advisor should select for analysis.

Managing the SQL Access Advisor

Using the `DBMS_ADVISOR` Package

1. Create and manage a task, by using a SQL workload object and a SQL Access task.
2. Specify task parameters, including workload and access parameters.
3. Using the workload object, gather the workload.
4. Using the SQL workload object and the SQL Access task, analyze the data.

You can also use the `QUICK_TUNE` procedure to quickly analyze a single SQL statement:

```
VARIABLE task_name VARCHAR2(255);
VARIABLE sql_stmt VARCHAR2(4000);
sql_stmt := 'SELECT COUNT(*) FROM customers
WHERE cust_region='TX'';
task_name := 'MY_QUICKTUNE_TASK';
DBMS_ADVISOR.QUICK_TUNE(DBMS_ADVISOR.SQLACCESS
_ADVISOR, task_name, sql_stmt);
```

Using the Database Control to Run the SQL Access Advisor

Under the **Performance** tab, click the **Advisor Central** link and then click the **SQL Access Advisor** link.

Note: Oracle creates the new indexes in the schema and tablespaces of the table on which they are created. If a user issues a query that leads to a recommendation to create a materialized view, Oracle creates the materialized view in that user's schema and tablespace.

Using the Performance Pages of the Database Control

The Database Home Page

Three major tuning areas the OEM Database Control will show you: CPU and wait classes, top SQL statements, and top sessions in the instance.

The Database Performance Page

This page shows the three main items:

Host

The Host part of the page shows two important graphs:

- **Average Run Queue:** This shows how hard the CPU is running.
- **Paging Rate:** This shows the rate at which the host server is writing memory pages to the swap area on disk.

Sessions waiting and working

The sessions graph shows which active sessions are on the CPU and which are waiting for resources like locks, disk I/O, and so on.

Instance throughput

If your instance throughput is decreasing, along with an increasing amount of contention within the database, you should start looking into tuning your database.

Indexing Enhancements

Skipping Unusable Indexes

In Oracle Database 10g, the `SKIP_UNUSABLE_INDEXES` parameter is an initialization parameter and its default value is `TRUE`. This setting disables error reporting of indexes and index partitions marked `UNUSABLE`.

Using Hash-Partitioned Global Indexes

- In Oracle 10g, you can create hash-partitioned global indexes.
- You can hash-partition indexes on tables, partitioned tables, and index-organized tables.
- This feature provides higher throughput for applications with large numbers of concurrent insertions.
- If you have queries with range predicates, for example, hash partitioned indexes perform better than range-partitioned indexes.

- You can't perform the following operations on hash-partitioned global indexes: ALTER INDEX REBUILD, ALTER TABLE SPLIT INDEX PARTITION, ALTER TABLE MERGE INDEX PARTITION, and ALTER INDEX MODIFY PARTITION.

```
CREATE INDEX sales_hash
on sales_items (sales_id) GLOBAL
PARTITION BY HASH (sales_id) (
  partition p1 tablespace tbs_1,
  partition p2 tablespace tbs_2,
  partition p3 tablespace tbs_3);

CREATE INDEX sales_hash
on sales_items (sales_id) GLOBAL
PARTITION BY HASH (sales_id)
partitions 4
store in (tbs_1,tbs_2,tbs_3,tbs_4);
```

Using the New UPDATE INDEXES Clause

Using the new UPDATE INDEXES clause during a partitioned table DDL command will help you do two things:

- specify storage attributes for the corresponding local index segments
- have Oracle automatically rebuild them.

```
ALTER TABLE MY_PARTS
MOVE PARTITION my_part1 TABLESPACE new_tbsp
UPDATE INDEXES
(my_parts_idx
(PARTITION my_part1 TABLESPACE my_tbsp))
```

Space and Storage Management Enhancements

Proactive Tablespace Management

- In Oracle Database 10g, by default, all tablespaces have built-in alerts that notify you when the free space in the tablespace goes below a certain predetermined threshold level.
- By default, Oracle sends out a warning alert when your tablespace is 85 percent full and a critical alert when the tablespace is 97 percent full.
- If you are migrating to Oracle Database 10g, Oracle turns off the automatic tablespace alerting mechanism by default.

Tablespace Alerts Limitations

- You can set alerts only for locally managed tablespaces.
- When you take a tablespace offline or make it read-only, you must turn the alerting mechanism off.
- You will get a maximum of only one undo alert during any 24-hour period.

Using the Database Control to Manage Thresholds
Manage **Metrics** link | click the **Edit Thresholds** button

Using the DBMS_SERVER_ALERT Package

You can use the procedures: SET_THRESHOLD and GET_THRESHOLD in the DBMS_SERVER_ALERT package to manage database thresholds.

Reclaiming Unused Space

In Oracle Database 10g, you can use the new segment-shrinking capability to make sparsely populated

segments give their space back to their parent tablespace.

Restrictions on Shrinking Segments

- You can only shrink segments that use Automatic Segment Space Management.
- You must enable row movement for heap-organized segments. By default, row movement is disabled at the segment level.

```
ALTER TABLE test ENABLE ROW MOVEMENT;
```

- You can't shrink the following:
 - Tables that are part of a cluster
 - Tables with LONG columns,
 - Certain types of materialized views
 - Certain types of IOTs.
 - Tables with function-based indexes.

Segment Shrinking Phases

There are two phases in a segment-shrinking operation:

Compaction phase

During this phase, the rows in a table are compacted and moved toward the left side of the segment and you can issue DML statements and queries on a segment while it is being shrunk.

Adjustment of the HWM/releasing space phase

During the second phase, Oracle lowers the HWM and releases the recovered free space under the old HWM to the parent tablespace. Oracle locks the object in an exclusive mode.

Manual Segment Shrinking

Manual Segment Shrinking is done by the statement:

```
ALTER TABLE test SHRINK SPACE;
```

You can shrink all the dependent segments as well:

```
ALTER TABLE test SHRINK SPACE CASCADE;
```

To only compact the space in the segment:

```
ALTER TABLE test SHRINK SPACE COMPACT;
```

Shrinking Segments Using the Database Control

To enable row movement:

Follow the links: **Schema, Tables, Edit Tables**, then **Options**.

To shrink a table segment:

Follow the links: **Schema, Tables**, select from the **Actions** field **Shrink Segments** and click **Go**.

Using the Segment Advisor

Choosing Candidate Objects for Shrinking

The Segment Advisor, to estimate future segment space needs, uses the growth trend report based on the AWR space-usage data.

Follow the links:

Database Home page, Advisor Central in the Related Links, Segment Advisor.

Estimating Object Size

You can use the Segment Advisor to determine your future segment resource usage.

follow these steps:

- From the Database Control home page, click the Administration tab.
- Under the Storage section, click the Tables link.
- Click the Create button to create a new table.
- You'll now be on the Create Table page. Under the Columns section, specify your column data types. Then click the Estimate Table Size button.

- On the Estimate Table Size page, specify the estimated number of rows in the new table, under Projected Row Count. Then click the Estimated Table Size button. This will show you the estimated table size.

Using the Undo and Redo Logfile Size Advisors

Undo Advisor

The Undo Advisor helps you perform the following tasks:

- Set the undo retention period
- Set the size of the undo tablespace

To access the Undo Advisor in the Database Control:

Follow the links: Database Control home page, **Administration**, **Undo Management** button, the **Undo Advisor** button in the right corner.

Redo Logfile Size Advisor

The Redo Logfile Size Advisor will make recommendations about the smallest online redo log files you can use.

The Redo Logfile Size Advisor is enabled only if you set the `FAST_START_MTTR_TARGET` parameter.

Check the column `OPTIMAL_LOGFILE_SIZE` in `V$INSTANCE_RECOVERY` view to obtain the optimal size of the redo log file for your `FAST_START_MTTR_TARGET` setting.

To access the Redo Logfile Size Advisor:

- Database Control home page, **Administration**, Under the **Storage** section, **Redo Log Groups**.
- Select any redo log group, and then choose the Sizing Advice option from the Action drop-down list, Click Go

Tablespace Enhancements

Managing the SYSAUX Tablespace

- Some Oracle features use SYSAUX in its operation.
- SYSAUX is mandatory in any database.
- SYSAUX cannot be dropped, renamed or transported.
- Oracle recommends that you create the SYSAUX tablespace with a minimum size of 240MB.

Creating SYSAUX

- DBCA creates it automatically and asks you about its configuration.
- Can be included in the manual database creation:

```
CREATE DATABASE mydb
USER SYS IDENTIFIED BY mysys
USER SYSTEM IDENTIFIED BY mysystem
..
SYSAUX DATAFILE 'c:\..\sysaux01.dbf' SIZE 500M
```

If you omit the `SYSAUX` clause, Oracle will create the `SYSAUX` tablespace automatically with their datafiles in location defined by the following rules:

- If you are using Oracle Managed Files (OMF), the location will be on the OMF.
- If OMF is not configured, default locations will be system-determined.
- If you include the `DATAFILE` clause for the `SYSTEM` tablespace, you must use the `DATAFILE` clause for the `SYSAUX` tablespace as well, unless you are using OMF.

You can use `ALTER TABLESPACE` command to add a datafile though.

Relocating SYSAUX Occupants

If there is a severe space pressure on the SYSAUX tablespace, you may decide to move components out of the SYSAUX tablespace to a different tablespace.

- Query the column `SPACE_USAGE_KBYTES` in the `V$SYSAUX_OCCUPANTS` to how much of the SYSAUX tablespace's space each of its occupants is currently using.
- Query the column `MOVE_PROCEDURE` to obtain the specific procedure you must use in order to move a given occupant out of the SYSAUX tablespace.

```
SQL> exec dbms_wm.move_proc('DRSYS');
```

Note: You can't relocate the following occupants of the SYSAUX tablespace: `STREAMS`, `STATSPACK`, `JOB_SCHEDULER`, `ORDIM`, `ORDIM/PLUGINS`, `ORDIM/SQLMM`, and `SMC`.

Renaming Tablespaces

In Oracle Database 10g, you can rename tablespaces: `ALTER TABLESPACE users RENAME TO users_new`

Restrictions:

- Your compatibility level must be set to 10.0 or higher.
- You can't rename the `SYSTEM` or `SYSAUX` tablespace, or offline tablespaces.
- If the tablespace is read-only, the datafile headers aren't updated, although the control file and the data dictionary are.

Renaming Undo Tablespace

- If database started using `init.ora` file, Oracle retrieves a message that you should set value of `UNDO_TABLESPACE` parameter.
- If database started using `spfile`, Oracle will automatically write the new name for the undo tablespace in your `spfile`.

Specifying the Default Permanent Tablespace During Database Creation

Use `DEFAULT TABLESPACE` clause in the `CREATE DATABASE` command

```
CREATE DATABASE mydb
...
DEFAULT TABLESPACE deftbls DATAFILE ...
```

If `DEFAULT TABLESPACE` not specified, `SYSTEM` tablespace will be used.

Note: The users `SYS`, `SYSTEM`, and `OUTLN` continue to use the `SYSTEM` tablespace as their default permanent tablespace.

After Database Creation Using SQL

Use `ALTER DATABASE` command as follows:

```
ALTER DATABASE DEFAULT TABLESPACE new_tbsp;
```

Using the Database Control

- Database Control home page, Administration, Storage Section, Tablespaces.
- Edit Tablespace page, select the Set As Default Permanent Tablespace option in the Type section. Then click Apply.

Viewing Default Tablespace Information

```
SELECT PROPERTY_VALUE FROM DATABASE_PROPERTIES
```

```
WHERE
PROPERTY_NAME='DEFAULT_PERMANENT_TABLESPACE'
```

Temporary Tablespace Groups

A temporary tablespace group is a list of temporary tablespaces.

Creating a Temporary Tablespace Group

You implicitly create a temporary tablespace group when you specify the `TABLESPACE GROUP` clause in a `CREATE TABLESPACE` statement:

```
CREATE TEMPORARY TABLESPACE temp_old TEMPFILE
'/u01/oracle/oradata/temp01.dbf' SIZE 500M
TABLESPACE GROUP group1;
```

You can also create a temporary tablespace group by:

```
ALTER TABLESPACE temp_old
TABLESPACE GROUP group1
```

Note: If you specify the `NULL` or `''` tablespace group, it is equivalent to the normal temporary tablespace creation statement (without any groups).

Setting a Group As the Default Temporary Tablespace for the Database

```
ALTER DATABASE DEFAULT TEMPORARY TABLESPACE
group1
```

Assigning a Temporary Tablespace Group to Users

```
CREATE USER sam IDENTIFIED BY sam
DEFAULT TABLESPACE users
TEMPORARY TABLESPACE group1;
ALTER USER SAM TEMPORARY TABLESPACE GROUP2;
```

Viewing Temporary Tablespace Group Information

Use the following views:

- o `DBA_TABLESPACE_GROUPS`
- o `DBA_USERS`

Bigfile Tablespaces

A bigfile tablespace (BFT) contains only one very large file (can be as large as 128 terabytes).

The main benefit is easier management of tablespaces and their datafiles.

Big File Tablespaces Restrictions

- You use bigfile tablespaces along with a Logical Volume Manager (LVM) or the Automatic Storage Management (ASM) feature, which support striping and mirroring.
- Both parallel query execution and RMAN backup parallelization would be adversely impacted, if you used bigfile tablespaces without striping.

Making Bigfile the Default Tablespace Type

Once you set the default type of your tablespace, all the tablespaces you subsequently create will be by default of the bigfile type:

```
CREATE DATABASE test
SET DEFAULT BIGFILE TABLESPACE ... ;
ALTER DATABASE SET DEFAULT BIGFILE TABLESPACE;
```

Creating a Bigfile Tablespace Explicitly

```
CREATE BIGFILE TABLESPACE bigtbs
DATAFILE '/u01/oracle/data/bigtbs_01.dbf' SIZE
100G ...
```

When you use the `BIGFILE` clause, Oracle will automatically create a locally managed tablespace with automatic segment-space management (ASSM).

You can use the keyword `SMALLFILE` in replacement with `BIGFILE` clause.

Altering a Bigfile Tablespace's Size

```
ALTER TABLESPACE bigtbs RESIZE 120G;
ALTER TABLESPACE bigtbs AUTOEXTEND ON NEXT
20G;
```

Viewing Bigfile Tablespace Information

All the following views have the new YES/NO column `BIGFILE`:

- o `DBA_TABLESPACES`
- o `USER_TABLESPACES`
- o `V$TABLESPACE`

Bigfile Tablespaces and ROWID Formats

	Bigfile tablespace	Smallfile tablespace
Format	Object# - Block# - Row#	Object# - File# - Block# - Row#
block number size	Can be much larger than smallfile tbs.	Is smaller than bigfile tbs.

For bigfile tablespaces, there is only a single file, with the relative file number always set to 1024.

The only supported way to extract the ROWID components is by using the `DBMS_ROWID` package.

You can specify the tablespace type by using the new parameter `TS_TYPE_IN`, which can take the values `BIGFILE` and `SMALLFILE`.

```
SELECT DISTINCT DBMS_ROWID.ROWID_RELATIVE_FNO
(rowid, 'BIGFILE ') FROM test_rowid
```

Using Sorted Hash Clusters

Sorted hash clusters are new data structures that allow faster retrieval of data for applications where data is consumed in the order in which it was inserted.

In a sorted hash cluster, the table's rows are already presorted by the sort key column.

Here are some of its main features:

- You can create indexes on sorted hash clusters.
- You must use the cost-based optimizer, with up-to-date statistics on the sorted hash cluster tables.
- You can insert row data into a sorted hash clustered table in any order, but Oracle recommends inserting them in the sort key column order, since it's much faster.

Creating Sorted Hash Cluster

```
CREATE CLUSTER call_cluster
(call_number NUMBER,
call_timestamp NUMBER SORT,
call_duration NUMBER SORT)
HASHKEYS 10000
SINGLE TABLE
HASH IS call_number
SIZE 50;
```

<code>SINGLE TABLE</code>	indicates that the cluster is a type of hash cluster containing only one table.
<code>HASH IS expr</code>	Specifies an expression to be used as the hash function for the hash cluster.
<code>HASHKEYS</code>	this clause creates a hash cluster and specify the number of hash values for the hash cluster.
<code>SIZE</code>	Specify the amount of space in bytes reserved to store all rows with the same cluster key value or the same hash value.

```
CREATE TABLE calls
```

```
(call_number NUMBER,
call_timestamp NUMBER,
call_duration NUMBER,
call_info VARCHAR2(50))
CLUSTER call_cluster
(call_number,call_timestamp,call_duration)
```

Copying Files Using the Database Server

The `DBMS_FILE_TRANSFER` package helps you copy binary files to a different location on the same server or transfer files between Oracle databases.

Both the source and destination files should be of the same type, either operating system files or ASM files.

The maximum file size is 2 terabytes, and the file must be in multiples of 512 bytes.

You can monitor the progress of all your file-copy operations using the `V$SESSION_LONGOPS` view.

Copying Files on a Local System

```
CREATE DIRECTORY source_dir AS
'/u01/app/oracle';

CREATE DIRECTORY dest_dir AS
'/u01/app/oracle/example';

BEGIN
  DBMS_FILE_TRANSFER.COPY_FILE(
    SOURCE_DIRECTORY_OBJECT => 'SOURCE_DIR',
    SOURCE_FILE_NAME => 'exm_old.txt',
    DESTINATION_DIRECTORY_OBJECT => 'DEST_DIR',
    DESTINATION_FILE_NAME => 'exm_new.txt');
END;
```

Transferring a File to a Different Database

```
BEGIN
  DBMS_FILE_TRANSFER.PUT_FILE(
    SOURCE_DIRECTORY_OBJECT => 'SOURCE_DIR',
    SOURCE_FILE_NAME => 'exm_old.txt',
    DESTINATION_DIRECTORY_OBJECT => 'DEST_DIR',
    DESTINATION_FILE_NAME => 'exm_new.txt',
    DESTINATION_DATABASE => 'US.ACME.COM');
END;
```

In order to transfer a file the other way around, you must replace the `PUT_FILE` procedure with the `GET_FILE` procedure.

The Oracle Scheduler and the Database Resource Manager

Simplifying Management Tasks Using the Scheduler

An Introduction to the Job Scheduler

- You may run PL/SQL and Java stored procedure, C functions, regular SQL scripts, and UNIX or Windows scripts.
- The Scheduler consists of the concepts: Program, Job, Schedule, Job class, Resource group, Window and Window Group.
- The Scheduler architecture consists primarily of the job table, job coordinator, and the job workers (or slaves).

Managing the Basic Scheduler Components

Creating Jobs

```
DBMS_SCHEDULER.CREATE_JOB(
  JOB_NAME => 'TEST_JOB1',
  JOB_TYPE => 'PLSQL_BLOCK',
  JOB_ACTION => 'DELETE FROM PERSONS WHERE
  SYSDATE=SYSDATE-1',
  START_DATE => '28-JUNE-04 07.00.00 PM
  AUSTRALIA/SYDNEY',
  REPEAT_INTERVAL => 'FREQ=DAILY;INTERVAL=2',
  END_DATE => '20-NOV-04 07.00.00 PM
  AUSTRALIA/SYDNEY',
  COMMENTS => 'TEST JOB')
```

JOB_TYPE Possible values are:

- o `plsql_block`
- o `stored_procedure`
- o `executable`

JOB_ACTION specifies the exact procedure, command, or script that the job will execute.

START_DATE and END_DATE These parameters specify the date that a new job should start and end. (Many jobs may not have an `end_date` parameter, since they are ongoing jobs.)

REPEAT_INTERVAL You can specify a repeat interval in one of two ways:

- o Use a PL/SQL date/time expression.
- o Use a database calendaring expression.

Specifying Intervals

FREQ takes YEARLY, MONTHLY, WEEKLY, DAILY, HOURLY, MINUTELY, and SECONDLY.

FREQ=DAILY; executes a job every 10 days
INTERVAL=10

FREQ=HOURLY; executes a job every other hour
INTERVAL=2

FREQ=WEEKLY; executes a job every Friday.
BYDAY=FRI

FREQ=WEEKLY; executes a job every other Friday.
INTERVAL=2;
BYDAY=FRI

FREQ=MONTHLY; executes a job on the last day of the month
BYMONTHDAY=1

FREQ=YEARLY; executes a job on the 31st of December.
BYMONTH=DEC;
BYMONTHDAY=31

FREQ=MONTHLY; executes a job every second Friday of the month
BYDAY=2FRI

Refer to PL/SQL Packages and Types Reference 10g Release 1, Chapter 83, Table 83-9 Values for `repeat_interval`.

Note: You'll be the owner of a job if you create it in your own schema. However, if you create it in another schema, that schema user will be owner of the job.

Enabling and Disabling Jobs

All jobs are disabled by default when you create them. You must explicitly enable them in order to activate and schedule them.

```
DBMS_SCHEDULER.ENABLE ('TEST_JOB1')
```

```
DBMS_SCHEDULER.DISABLE ('TEST_JOB1')
```

Dropping a Job

```
DBMS_SCHEDULER.DROP_JOB (JOB_NAME =>
'test_job1')
```

Running and Stopping a Job

```
DBMS_SCHEDULER.RUN_JOB('TEST_JOB1')
DBMS_SCHEDULER.STOP_JOB('TEST_JOB1')
```

In both the `STOP_JOB` and `RUN_JOB` procedures, there is a `FORCE` argument, which is set to `FALSE` by default. By setting `FORCE=TRUE`, you can stop or drop a job immediately by using the appropriate procedure. You must have the `MANAGE_SCHEDULER` system privilege to use the `FORCE` setting.

Creating a Program

```
DBMS_SCHEDULER.CREATE_PROGRAM(
  PROGRAM_NAME => 'TEST_PROGRAM',
  PROGRAM_ACTION => 'SCOTT.UPDATE_SCHEMA_STATS',
  PROGRAM_TYPE => 'STORED_PROCEDURE',
  ENABLED => TRUE)
```

Note: If you want to create the program in a different user's schema, you must qualify the program name with the schema name.

`TEST_JOB1` job can then be created using the program component as follows:

```
DBMS_SCHEDULER.CREATE_JOB(
  JOB_NAME => 'TEST_JOB1',
  PROGRAM_NAME => 'TEST_PROGRAM',
  REPEAT_INTERVAL=> 'FREQ=DAILY;BYHOUR=12',
  ENABLED => TRUE)
```

Enabling and Disabling Programs

```
DBMS_SCHEDULER.ENABLE('TEST_PROGRAM')
DBMS_SCHEDULER.DISABLE('TEST_PROGRAM')
```

Dropping a Program

```
DBMS_SCHEDULER.DROP_PROGRAM('TEST_PROGRAM')
```

Creating a Schedule

```
DBMS_SCHEDULER.CREATE_SCHEDULE(
  SCHEDULE_NAME => 'TEST_SCHEDULE',
  START_DATE => SYSTIMESTAMP,
  END_DATE => SYSTIMESTAMP + 30,
  REPEAT_INTERVAL => 'FREQ=HOURLY;INTERVAL= 12',
  COMMENTS => 'EVERY 12 HOURS')
```

Note the following about creating a Schedule:

- When you create a schedule, Oracle provides access to `PUBLIC`. Thus, all users can use your schedule, without any explicit grant of privileges to do so.
- You specify the start and end times using the `TIMESTAMP WITH TIME ZONE` datatype. The Scheduler also supports all `NLS_TIMESTAMP_TZ_FORMAT` settings.
- You must use a calendaring expression to create the repeat interval.

```
DBMS_SCHEDULER.CREATE_JOB(
  JOB_NAME => 'TEST_JOB02',
  PROGRAM_NAME => 'TEST_PROGRAM',
  SCHEDULE_NAME => 'TEST_SCHEDULE')
```

Altering a Schedule

You can alter the attributes (except `SCHEDULE_NAME`) of a schedule by using the `SET_ATTRIBUTE` procedure of the `DBMS_SCHEDULER` package.

Dropping a Schedule

```
DBMS_SCHEDULER.DROP_SCHEDULE(SCHEDULE_NAME =>
  'TEST_SCHEDULE');
```

When you drop a schedule by using the `FORCE=TRUE` attribute, you'll drop the schedule, even if there are jobs and windows that use the schedule. The Scheduler first

disables the dependent jobs/windows before dropping the schedule itself.

Managing Advanced Scheduler Components

Creating a Job Class

- Using job classes helps you prioritize jobs by allocating resources differently among the various jobs.
- All job classes are created in the `SYS` schema. To create a job class you need `MANAGE_SCHEDULER` privilege.
- For users to create jobs that belong to a job class, the job owner must have `EXECUTE` privileges on the job class.
- There is a default job class, `DEFAULT_JOB_CLASS`, to which all jobs will belong if you don't explicitly assign them to a job class.

```
DBMS_SCHEDULER.CREATE_JOB_CLASS (
  JOB_CLASS_NAME => 'ADMIN_JOBS',
  RESOURCE_CONSUMER_GROUP => 'ADMIN_GROUP',
  LOGGING_LEVEL => DBMS_SCHEDULER.LOGGING_OFF
  LOGGING_HISTORY => 30,
  COMMENTS => 'Admin related jobs.')
```

`LOGGING_LEVEL` This attribute specifies how much information is logged. The three possible options are:

- `DBMS_SCHEDULER.LOGGING_OFF`
- `DBMS_SCHEDULER.LOGGING_RUNS`
- `DBMS_SCHEDULER.LOGGING_FULL`:

In addition to recording every run of a job, the Scheduler will log every time a job is created, enabled, disabled, altered, and so on.

Note: As a DBA, you can set logging at the job class level in order to audit Scheduler jobs. In this case, an individual user can only increase the amount of logging the individual job level.

`LOGGING_HISTORY` Specifies the number of days (default is 30) that the database will retain the logs before purging them.

Oracle will automatically create a daily job called the `PURGE_LOG`, which cleans the log entries.

Manually Purging a Job Class Log

By default, once a day, the Scheduler will purge all window logs and job logs that are older than 30 days.

```
DBMS_SCHEDULER.PURGE_LOG(LOG_HISTORY=7,
  WHICH_LOG =>'JOB_LOG')
```

`LOG_HISTORY` This specifies how much history (in days) to keep. The valid range is 0-999. If set to 0, no history is kept.

`WHICH_LOG` This specifies which type of log. Valid values are: `JOB_LOG`, `WINDOW_LOG`, and `JOB_AND_WINDOW_LOG`.

You can purge log of a specific job:

```
DBMS_SCHEDULER.PURGE_LOG (
  LOG_HISTORY => 1,
  JOB_NAME => 'TEST_JOB1')
```

You can modify the retention period (the default is 30days) of the logs for a job class:

```
DBMS_SCHEDULER.SET_ATTRIBUTE (
'TEST_JOB_CLASS', 'log_history', '7')
```

In order to clear all window and job logs:

```
DBMS_SCHEDULER.PURGE_LOG()
```

Altering a Job Class

```
DBMS_SCHEDULER.SET_ATTRIBUTE (
NAME => 'ADMIN_JOBS',
ATTRIBUTE => 'START_DATE',
VALUE => '01-JAN-2005 9:00:00 PM US/Pacific')
```

You can change the `START_DATE`, `END_DATE`, and other logging-related attributes as well.

Dropping a Job Class

```
DBMS_SCHEDULER.DROP_JOB_CLASS('TEST_CLASS')
```

If you want to drop a job class with jobs in it, you must specify the `FORCE=TRUE` option in your `DROP_JOB_CLASS` procedure. When you do this, the jobs in the dropped job class are disabled and moved to the default job class in your database. If the job is already running when you drop its job class, the job will run to completion anyway.

Working with Scheduler Windows

Windows enable the automatic changing of resource plans based on a schedule.

Creating a Window

- Windows are always created in the `SYS` schema.
- To create a window, you must have the `MANAGE_SCHEDULER` system privilege.
- A window is automatically enabled upon its creation.

```
DBMS_SCHEDULER.CREATE_WINDOW (
WINDOW_NAME => 'TEST_WINDOW',
START_DATE => '01-JAN-05 12:00:00AM',
REPEAT_INTERVAL => 'FREQ=DAILY',
RESOURCE_PLAN => 'TEST_RESOURCEPLAN',
DURATION => INTERVAL '60' MINUTE,
END_DATE => '31-DEC-05 12:00:00AM',
WINDOW_PRIORITY => 'HIGH',
COMMENTS => 'Test Window')
```

<code>START_DATE</code>	Time when the Window will open.
<code>REPEAT_INTERVAL</code>	The next time the window will open again.
<code>RESOURCE_PLAN</code>	Tells us that while this window is open, resource allocation to all the jobs that run in this window will be guided by the resource plan directives in the resource plan <code>TEST_RESOURCEPLAN</code> .
<code>DURATION</code>	Window will remain open for a period of 60 minutes, after which it will close.
<code>END_DATE</code>	Window will open for the last time on December 31, 2005, after which it will be disabled and closed.
<code>WINDOW_PRIORITY</code>	Possible values are: <code>LOW</code> , <code>HIGH</code> . When two Windows overlap, the high-priority window will open and the lower-priority window does not open.

You can create a window using a saved schedule:

```
DBMS_SCHEDULER.CREATE_WINDOW (
WINDOW_NAME => 'TEST_WINDOW',
```

```
SCHEDULE_NAME => 'TEST_SCHEDULE',
RESOURCE_PLAN => 'TEST_RESOURCEPLAN',
DURATION => interval '160' minute,
COMMENTS => 'Test Window')
```

Opening a Window

- A window will automatically open at a time specified by its `START_TIME` attribute.
- Only one window can be open at any given time.
- A window can be manually opened:

```
DBMS_SCHEDULER.OPEN_WINDOW (
WINDOW_NAME =>'BACKUP_WINDOW',
DURATION => '0 12:00:00')
```

When you specify the duration, you can specify days, hours, minutes, seconds, in that order.

- You can open an already open window. If you do this, the duration of the window will last a time period equal to its duration attribute.

Closing a Window

```
DBMS_SCHEDULER.CLOSE_WINDOW('BACKUP_WINDOW')
```

A running job may close upon the closing of its window, if you create a job with the attribute `STOP_ON_WINDOW_CLOSE` set to `TRUE`.

Disabling a Window

- You can only disable a window if no job uses that window or if the window is not open.
- If the window is open, you can disable it by using the `DISABLE` program with the `FORCE=TRUE` attribute.

```
DBMS_SCHEDULER.DISABLE (NAME =>
'BACKUP_WINDOW')
```

Dropping a Window

- You can drop a window by using the `DROP_WINDOW` procedure.
- If a job associated with a window is running, a `DROP_WINDOW` procedure will continue to run through to completion and is disabled after it completes.
- If you set the `STOP_ON_WINDOW_CLOSE` attribute to `TRUE`, however, the job will immediately stop when you drop an associated window.

Prioritizing Jobs

- You can prioritize jobs at two levels: *class* and *job*.
- The prioritization at the class level is based on the resources allocated to each resource consumer group by the currently active resource plan. The consumer group that a job class maps to can be specified when creating a job class.
- At job level, the job priority ranges from 1 to 5, with 1 being the highest priority and 3 being the default.
- When you have more than one job within the same class scheduled for the same time, the `JOB_PRIORITY` of the individual jobs determines which job starts first.

```
DBMS_SCHEDULER.SET_ATTRIBUTE (
NAME => 'test_job',
ATTRIBUTE => 'job_priority',
VALUE => 1)
```

Window Priorities

If there are more than one window to open at the same time, the Scheduler will close all windows except one, using the following rules of precedence:

- If two windows overlap, the window with the higher priority opens and the window with the lower priority closes.
- If two windows of the same priority overlap, the active window remains open.
- If you are at the end of a window and you have other windows defined for the same time period, the window that has the highest percentage of time remaining will open.

Window Groups

- A window group is a collection of windows, and is part of the SYS schema.
- The concept of a window group is for convenience only, and its use is purely optional.

Unsetting Component Attributes

```
DBMS_SCHEDULER.SET_ATTRIBUTE_NULL('test_program', 'COMMENTS')
```

Altering Common Component Attributes

- There are some attributes that are common to all Scheduler components.
- Use the procedure `SET_SCHEDULER_ATTRIBUTE` to set these common, or global level, attributes.
- These are the global attributes:

`DEFAULT_TIMEZONE`

If jobs and windows specifications use the calendaring syntax but omit the start date, the Scheduler derives the time zone from the `DEFAULT_TIMEZONE` attribute.

Oracle recommends that you set the `DEFAULT_TIMEZONE` attribute to a region's name instead of absolute time zone offset, in order to ensure that daylight saving adjustments are being taken into account.

`LOG_HISTORY`

This attribute refers to the number of days the Scheduler will retain job and window logs.

`MAX_JOB_SLAVE_PROCESSES`

The Scheduler determines the optimal number of job slave processes, based on your processing requirements. However, you can set a limit on the number of job slave processes using the `MAX_JOB_SLAVE_PROCESSES` attribute, whose default value is NULL, and the range is from 1 to 999.

Viewing Information About the Scheduler

`DBA_SCHEDULER_JOBS` This view provides the status and general information about scheduled jobs in your database.

`DBA_SCHEDULER_RUNNING_JOBS` This view provides you with information regarding currently running jobs.

`DBA_SCHEDULER_JOB_RUN_DETAILS` This view provides information about status and the duration of execution for all jobs in your database.

`DBA_SCHEDULER_SCHEDULES` This view provides information on all current schedules in your

database.

Database Resource Manager Enhancements

Setting Idle Time-Outs

You can now limit the maximum idle time for a session as well as the maximum time an idle session can block another session.

```
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE
(PLAN => 'NEW_PLAN',
 GROUP_OR_SUBPLAN => 'SALES',
 COMMENT => 'SALES GROUP', CPU_P1 => 60,
 PARALLEL_DEGREE_LIMIT_P1_P1 => 4
 MAX_IDLE_TIME => 600,
 MAX_IDLE_BLOCKER_TIME => 300)
```

Automatic Switching Back to Initial Consumer Groups

When you create plan directives for the Database Resource Manager using the `CREATE_PLAN_DIRECTIVE` procedure of the `DBMS_RESOURCE_MANAGER` package, you can specify the following parameters:

`SWITCH_TIME`

Specifies the time (in seconds) that a session can execute before an action is taken. Default is UNLIMITED.

`SWITCH_TIME_IN_CALL`

Specifies the time (in seconds) that a session can execute before an action is taken. At the end of the call, the consumer group of the session is restored to its original consumer group. Default is UNLIMITED.

Note: You cannot specify both `SWITCH_TIME` and `SWITCH_TIME_IN_CALL`.

Mappings to Assign Priorities to Resource Groups

You set session attribute mapping priorities by using the `SET_CONSUMER_GROUP_MAPPING_PRI` procedure of the `DBMS_RESOURCE_MANAGER` package.

```
DBMS_RESOURCE_MANAGER.SET_CONSUMER_GROUP_MAPPING_PRI (
(EXPLICIT => 1, CLIENT_MACHINE => 2,
 MODULE_NAME => 3, ORACLE_USER => 4,
 SERVICE_NAME => 5, CLIENT_OS_USER => 6,
 CLIENT_PROGRAM => 7, MODULE_NAME_ACTION => 8,
 SERVICE_MODULE => 9,
 SERVICE_MODULE_ACTION => 10)
```

Note: Application developers may also set the `MODULE_NAME` and `MODULE_NAME_ACTION` through the use of the `DBMS_APPLICATION_INFO` package. The `SERVICE_NAME` attribute is the connect string that you specify in your `tnsnames.ora` file.

New Database Resource Manager Allocation Methods

The `RUN_TO_COMPLETION` Allocation Method

When you create a consumer group using the `CREATE_CONSUMER_GROUP` procedure, the `CPU_MTH` option provides the method to distribute your CPU among the sessions in the consumer group. The default value for the `CPU_MTH` option is `ROUND_ROBIN`. The new `RUN_TO_COMPLETION` method specifies that the session with the largest active time should be scheduled ahead of other sessions.

```
DBMS_RESOURCE_MANAGER.CREATE_CONSUMER_GROUP (
CONSUMER_GROUP => 'SALES', CPU_MTH => 'RUN TO
COMPLETION')
```

The RATIO Allocation Method

The RATIO allocation method is meant for single-level resource plans that use ratios to specify the allocation of CPU.

The RATIO and the old EMPHASIS allocation methods are used with the CREATE_PLAN procedure and apply to resource plans. Then You must also use the CREATE_PLAN_DIRECTIVE procedure and set the CPU_P1 directive to actually set the ratios for the CPU allocation.

```
DBMS_RESOURCE_MANAGER.CREATE_PLAN
(PPLAN => 'SERVICE_LEVEL_PLAN',
CPU_MTH -> 'RATIO',
COMMENT => 'SERVICE LEVEL PLAN');
```

```
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE
(PPLAN => 'SERVICE_LEVEL_PLAN',
GROUP_OR_SUBPLAN => 'GOLD_CG',
COMMENT => 'GOLD SERVICE LEVEL CUSTOMERS',
CPU_P1 => 10);
```

... and so on to other groups.

Backup and Recovery Enhancements

Using the Flash Recovery Area

The flash recovery area serves as the default storage area for all files related to backup and restore operations.

The flash recovery area provides the following benefits:

- Single storage location for all recovery-related files.
- Automatic management of recovery-related disk space.
- Faster backup and restore operations, since you don't need to restore tape backups.
- Increased reliability of backups, since disks are generally safer storage devices than tapes.

What's in the Flash Recovery Area?

The flash recovery area may contain the following files:

- **Datafile copies:** The new RMAN command BACKUP AS COPY can be used to create image copies of all datafiles and automatically store in the flash recovery area.
- **Control file autobackups:** The database places any control file backups it generates in the flash recovery area.
- **Archived redo log files:** If you store Archived redo log files in the flash recovery area, Oracle will automatically delete the files.
- **Online redo log files:** Oracle recommends that you save a multiplexed copy of your online redo log files in the flash recovery area. The following statements can create online redo logs in the flash recovery area:
CREATE DATABASE, ALTER DATABASE ADD LOGFILE, ALTER DATABASE ADD STANDBY LOGFILE, and ALTER DATABASE OPEN RESETLOGS.
- **Current control files:** Oracle also recommends that you store a multiplexed current control file in the flash recovery area.
- **RMAN files**
- **Flashback logs:** If you enable the flashback database feature, Oracle copies images of each

altered block in every datafile into flashback logs stored in the flash recovery area.

Note: Oracle calls the multiplexed redo log files and control files in the flash recovery area **permanent files**, since they should never be deleted and are part of the live database. Oracle terms all the other files in the flash recovery area (recovery related files) **transient files**, since Oracle will delete them eventually after they have become obsolete or have already been copied to tape.

Creating a Flash Recovery Area

You use the DB_RECOVERY_FILE_DEST and DB_RECOVERY_FILE_DEST_SIZE initialization parameters to configure a flash recovery area in your database.

When you use the DB_RECOVERY_FILE_DEST parameter to specify the destination of your flash recovery area, you can use a directory, file system, or ASM disk group as your destination.

Dynamically Defining the Flash Recovery Area

```
ALTER SYSTEM SET DB_RECOVERY_FILE_DEST_SIZE =
2G SCOPE=BOTH
```

```
ALTER SYSTEM SET DB_RECOVERY_FILE_DEST =
'C:\ORACLE\RECOVERY_AREA' SCOPE=BOTH
```

You must always specify the size parameter before specifying the location parameter.

Disabling the Current Flash Recovery Area

```
ALTER SYSTEM SET DB_RECOVERY_FILE_DEST = ''
```

Note: even after you disable the flash recovery area, the RMAN will continue to access the files located in the flash recovery area for backup and recovery purposes.

Default File Location and the Flash Recovery Area

The initialization parameters DB_CREATE_FILE_DEST and DB_CREATE_ONLINE_LOG_DEST_n determine the location of all OMF files.

Control Files

If you haven't set the CONTROL_FILES parameter, Oracle will create the control files in various default locations, according to the following rules:

- If you specify the DB_CREATE_ONLINE_LOG_DEST_n parameter, Oracle will create an OMF-based control file in n number of locations, with the first directory holding the primary control file.
- If you specify the DB_CREATE_FILE_DEST and DB_RECOVERY_FILE_DEST parameters, Oracle will create an OMF based control file in both of these locations.
- If you just specify the DB_RECOVERY_FILE_DEST parameter, Oracle will create an OMF-based control file in the flash recovery area only.
- If you omit all three of the initialization parameters, Oracle will create a non-OMF-based control file in the system-specific default location.

Note: If the database creates an OMF control file, and it is using a server parameter file, then the database sets the CONTROL_FILES initialization parameter in the server parameter file.

Redo Log Files

If you omit the LOGFILE clause during database creation, Oracle will create the redo log files according to the same rules as mentioned above.

Backing Up the Flash Recovery Area

In order to back up the flash recovery area itself using RMAN, you must set `CONFIGURE BACKUP OPTIMIZATION` to `ON`.

You can back up the flash recovery area only to a tape device using these backup commands:

BACKUP RECOVERY AREA

- This command backs up all flash recovery files in the current or previous flash recovery area destinations.
- It backs up only those files that have never been backed up to tape before.
- The files that the command will back up include full backups, incremental backups, control file autobackups, archive logs, and datafile copies.

BACKUP RECOVERY FILES

This command backs up all the files that the `BACKUP RECOVERY AREA` command does, but from all areas on your file system, not just from the flash recovery area.

BACKUP RECOVERY FILE DESTINATION

Use this command to move disk backups created in the flash recovery area to tape.

Note: Neither of the two commands, `BACKUP RECOVERY AREA` or `BACKUP RECOVERY FILES`, will back up any permanent files or the flashback logs in the flash recovery area.

Managing the Flash Recovery Area

Space Management

If you ever receive the out-of-space warning (85) and critical alerts (97) because of space pressure in your flash recovery area, you have the following options:

- Consider changing your backup retention and archive log retention policies.
- Increase the size of the `DB_RECOVERY_FILE_DEST_SIZE` parameter to allocate more space to your current flash recovery area.
- Use the `BACKUP RECOVERY AREA` command in the RMAN to back up the contents of the flash recovery area to a tape device.
- Use the RMAN to delete unnecessary backup files. The RMAN commands `CROSSCHECK` and `DELETE EXPIRED` come in handy during this deletion process.

Data Dictionary Views

V\$RECOVERY_FILE_DEST

This view is the main source and contains the following columns:

<code>SPACE_LIMIT</code>	how much space has been allocated to the flash recovery area
<code>SPACE_USED</code>	space occupied
<code>SPACE_RECLAIMABLE</code>	how much space you can reclaim by getting rid of obsolete and redundant files in the flash recovery area.
<code>NUMBER_OF_FILES</code>	number of files

Moving the Flash Recovery Area

```
ALTER SYSTEM SET
DB_RECOVERY_FILE_DEST='/u01/app/oracle/new_area'
SCOPE=BOTH
```

Eventually, Oracle will delete all the transient files from the previous flash recovery area location, when each of them become eligible for deletion. However, if you want to move your current permanent files, transient files, or flashback logs to the new flash recovery area, you can do so by using the standard file-moving procedures.

Using Incremental Backups

Recovering with Incrementally Updated Backups

You can apply incremental backups to your datafile image copies when you use the RMAN.

Here are the steps you follow:

1. Apply the incremental backups to datafile image copies.
2. Then apply the archive logs since the last incremental backup only.

Note: Archived redo logs apply changes at the transaction level. Incremental backups restore datafiles by applying changes at the database block level (faster).

Fast Incremental Backups

- RMAN reads change tracking file to find out which data blocks to read and copy during an incremental backup process, to avoid needing to read entire datafiles during incremental backups.
- A new background process, the change tracking writer (CTWR), is in charge of writing the block change information to the change tracking file.

Change Tracking File Features

- The change tracking file contains the physical location of all database changes.
- The minimum size is 10MB. Oracle creates the file automatically and allocates additional space in 10MB increments.
- The file's size depends on your database size, number of instances in an RAC, and the number of old backups the file maintains.
- `V$BLOCK_CHANGE_TRACKING` shows the name, size, and status of your change tracking file.

Enabling Block Change Tracking

```
ALTER DATABASE
ENABLE BLOCK CHANGE TRACKING
USING FILE
'C:\ORACLE\RECOVERY_AREA\CHANGETRACK.LOG'
```

To relocate the file, while in mount stage:

```
ALTER DATABASE RENAME FILE
'C:\ORACLE\RECOVERY_AREA\CHANGETRACK.LOG'
TO 'C:\ORACLE\NEWCHANGE.LOG'
```

To disable the file:

```
ALTER DATABASE DISABLE BLOCK CHANGE TRACKING;
```

Enhanced RMAN Commands

Using the BACKUP AS COPY Command

- The RMAN `COPY` command has been deprecated in Oracle Database 10g and replaced with `BACKUP AS COPY` command.
- `BACKUP AS COPY` command enables you to copy: database, tablespaces, datafiles, archived redo logs and control files.

- If you want RMAN to create image copies by default (rather than backupset):

```
RMAN> configure device type disk backup type to copy
```

- To create a backup set in the command level:

```
RMAN> backup as backupset database
```

Performing Backups

```
RMAN> backup database;
RMAN> backup copy of database;
RMAN> backup tablespace users;
RMAN> backup copy of tablespace users;
RMAN> backup datafile 10;
RMAN> backup copy of datafile 10;
RMAN> backup current controlfile;
RMAN> backup controlfilecopy all;
```

Using the CATALOG Command

```
RMAN> catalog backuppiece 'filename'
RMAN> catalog datafilecopy 'filename'
RMAN> change backuppiece 'file_name' uncatalog
```

Using the CATALOG START WITH Command

You can ask the RMAN to search in a certain directory for all backup files that aren't part of the catalog already:

```
RMAN> catalog start with
"C:\ORACLE\FLASH_RECOVERY_AREA\NINA\DATAFILE"
```

Enhancements in RMAN Scripts

1. Convertibility of RMAN Scripts

In Oracle Database 10g, you can change a stored script into a text script and vice versa.

```
RMAN> print script full_backup to file
'my_script_file.txt'
```

2. Global RMAN Scripts

Oracle Database 10g provides a new concept of global scripts, which you can execute against any database registered in the recovery catalog, as long as your RMAN client is connected to the recovery catalog and a target database simultaneously.

```
RMAN> create global script global_full_backup
{ backup database plus archivelog;
  delete obsolete; }
```

Miscellaneous Backup and Recovery Enhancements

Using the Database Control to Configure Backups

On the Database Control home page-> Maintenance tab-> Configure Backup Settings.

You can use one of the following choices to tell RMAN where to place its target files:

- FORMAT option in a backup command
- CONFIGURE CHANNEL FORMAT option
- DB_RECOVERY_FILE_DEST

Automatic Channel Failover

If one of the channels on which the RMAN is performing a backup fails, the RMAN will automatically try to use an

alternative channel, provided you are using multiple channels.

Implementing Fast Recovery

For those special times when you really need a fast recovery, Oracle Database 10g offers the SWITCH DATABASE command.

The RMAN simply adjusts the pointers for the datafiles in the control file, so they now point to the backup files in your flash recovery area.

```
RMAN> SWITCH DATABASE TO COPY
```

Recovering Datafiles Without Backups

The ability to recover a file that has never been backed up has always been available from SQL*Plus, with the help of the CREATE DATAFILE X AS Y statement. Now, in Oracle Database 10g, you can create the lost file as part of an RMAN RESTORE DATABASE command.

Compressed Backups

- Oracle Database 10g lets you compress RMAN backups to save on storage.
- You must set the COMPATIBLE initialization parameter to a minimum of 10.0.0.
- You can't compress an image copy; you can compress a backup only if you are using backup sets.
- The V\$BACKUP_FILES view contains information on backup files including the compression status.

```
RMAN> CONFIGURE DEVICE TYPE DISK BACKUP TYPE TO COMPRESSED BACKUPSET
```

```
RMAN> BACKUP AS COMPRESSED BACKUPSET DATABASE
```

Simplified Recovery Through RESETLOGS

- In Oracle Database 10g, you can use backups taken before an incomplete recovery operation; that is, you can use backups from older incarnations of the database.
- The new archive redo log format in Oracle Database 10g is of the following form:
LOG_ARCHIVE_FORMAT="log%t_%s_%r.arc"
The additional variable r stands for the RESETLOGS identifier.
- The V\$DATABASE view contains now RESETLOGS_CHANGE#, RESETLOGS_TIME, and RESETLOGS_ID.
- The V\$LOG_HISTORY contains now RESETLOGS_CHANGE# and RESETLOGS.

Dropping a Database

Here are some features of the DROP DATABASE command:

- Oracle will drop all control files and datafiles automatically, whether you use the SQL*Plus, RMAN, or DBCA interface to drop a database.
- Oracle doesn't remove archived redo logs and backups. To make the RMAN remove all database backup copies as well as the archived redo log files:
RMAN> DROP DATABASE INCLUDING BACKUPS
- If you are using SPFILE, Oracle will remove it automatically.

- After you drop a database, the RMAN catalog continues to list the dropped database information. You need to use the following RMAN command:

```
RMAN> UNREGISTER DATABASE
```

Specifying Limits for Backup Duration

You can use the `DURATION` command as an option for your regular backup commands, such as `BACKUP AS COPY`, to specify the time (in hours and minutes) a backup job can take. This makes the job taken less resources during its operation.

```
DURATION <hrs>:<mins> [PARTIAL] [MINIMIZE {TIME|LOAD}]
```

PARTIAL

Normally, when your database backup jobs run past the time interval you specify through the `DURATION` parameter, the RMAN job errors out and the backup is canceled. You can override this default behavior by specifying the `PARTIAL` clause, which will prevent the issuing of any RMAN error messages.

MINIMIZE TIME

This option tells the RMAN to “hurry up” and finish as fast as it can.

MINIMIZE LOAD

This option tells the RMAN to “slow down” if it is well within its allotted time for a backup job.

Note: It is recommended that you do not use the `MINIMIZE LOAD` option with tape.

Placing All Datafiles in Backup Mode

Starting the Online Backup

You can use the `ALTER DATABASE` command to specify that all datafiles in the entire database be placed in the backup mode simultaneously.

Ending the Online Backup

You must use the `ALTER DATABASE END BACKUP` command to end the online backup. You can issue this command when the database is in a mounted or an open state.

Automatic Auxiliary Instance Creation

When you perform a tablespace point-in-time recovery (TSPITR) to recover from certain database errors, Oracle Database 10g will now automatically create the auxiliary instance and remove it after the recovery is over.

You must provide the locations for all files of the auxiliary instance to the RMAN, so it can create the instance.

Flashback Technology Enhancements

Using the Flashback Technology

You can use the flashback technology at the database, table, and transaction levels:

- **Flashback database** enables you to take the entire database to a past point in time (using flashback logs).
- **Flashback drop** lets you retrieve accidentally dropped tables and indexes (using the recycle bin).
- **Flashback table** lets you recover a table to a time in the past (using undo data).

- **Flashback query** lets you query and restore data rows to a point in time (using undo data).

General Flashback Technology

Guaranteed Undo Retention

The initialization parameter `UNDO_RETENTION` enables you to specify the length of time Oracle must retain undo information in the undo segments.

Default value: 900 (in seconds)

Modifiable: ALTER SYSTEM

Range: 0 to $2^{32} - 1$

By default, Oracle doesn’t guarantee undo retention.

Methods to specify Undo Guarantee:

- By using the `RETENTION GUARANTEE` clause when you create an undo tablespace:

```
CREATE UNDO TABLESPACE test1
DATAFILE 'c:\oracle\oradata\undotbs_01.dbf'
SIZE 100M AUTOEXTEND ON
RETENTION GUARANTEE
```

- You can also use the `ALTER TABLESPACE` command:

```
ALTER TABLESPACE test1 RETENTION GUARANTEE
```

- You can specify undo guarantee for the undo tablespace when you create the database.

Note: You can use the `RETENTION NOGUARANTEE` clause to turn off the guaranteed retention of undo information.

Note: The amount of time for which undo is retained for the Oracle Database for the current undo tablespace can be obtained by querying the `TUNED_UNDORETENTION` column of the `V$UNDOSTAT` dynamic performance view.

Note: Use Oracle’s Undo Advisor to get approximate undo parameter values as well as suggestions regarding the sizing of your undo tablespace to successfully support flashback for a specified time.

Time Mapping Granularity

- Oracle gives you a choice between using either clock time or the system change number (SCN) to specify exactly what time in the past you want to go back to.
- The `SCN_TO_TIMESTAMP` SQL function lets you convert an SCN to a calendar time (`TIMESTAMP`) value. There is a mapping granularity of three seconds.

```
SELECT current_scn,
SCN_TO_TIMESTAMP(current_scn) FROM v$database
```
- The `TIMESTAMP_TO_SCN` function converts a timestamp to its corresponding SCN.

Flashback Database

How Flashback Database Works

- Once you enable the flashback database feature, at regular intervals, a new process RVWR (RecoveryWriter) copies images of each altered block in the datafiles from memory (*flashback buffer*) to the new *flashback logs*.
- Oracle stores these flashback logs in the flashback recovery area.
- If you want to flashback to 8:00 A.M., it may turn out that the flashback logs nearest to the target time were written at 7:56 A.M. To cover this gap, you must

apply the changes from archived or online redo log files pertaining to that period.

- Always remember that Oracle doesn't guarantee that you can flashback your database to the flashback retention target. If Oracle is running low on free space in the flash recovery area for newly arriving archived redo log files, it will remove some flashback logs to make room.

Flashback Database Considerations

- If a datafile was resized during the time span covered by the Flashback Database operation, you can't flashback that datafile. Instead, you must offline that particular datafile before you start the flashback database operation.
- If a control file has been restored or re-created during the time span you want to flashback over, you can't use the Flashback Database feature.
- You can't flashback a database to before a RESETLOGS operation.
- You can't flashback a datafile that was dropped or shrunk during the time span covered by the flashback table operation.

Configuring Flashback Database

1. Ensure that your database is in the archive log mode.

```
V$DATABASE (cols: logmode)
ARCHIVE LOG LIST (in SQL*Plus)
```

2. Your database must be using the flash recovery area.

```
SELECT VALUE FROM V$PARAMETER WHERE NAME =
'db_recovery_file_dest'
```

3. You must set the initialization parameter `DB_FLASHBACK_RETENTION_TARGET` to set your flashback retention target (in minutes).

```
ALTER SYSTEM SET
DB_FLASHBACK_RETENTION_TARGET=1440
```

4. Shut down the database and restart in the `MOUNT EXCLUSIVE` mode.

5. Turn the flashback database feature on with the following command:

```
ALTER DATABASE FLASHBACK ON;
```

6. Use the `ALTER DATABASE OPEN` command to open the database.

```
SELECT FLASHBACK_ON FROM V$DATABASE;
```

Note: You can turn the feature off by using the `ALTER DATABASE FLASHBACK OFF` command while the database is in the `MOUNT EXCLUSIVE` mode. When you do so, Oracle deletes all flashback database logs in the flash recovery area.

Note: If you don't want certain tablespaces to be part of your flashback operations, issue the following command after setting the tablespace offline:

```
ALTER TABLESPACE USERS FLASHBACK OFF
```

Flashbacking a Database

1. Restart the database in the `MOUNT (exclusive)` mode then issue one of the commands:

```
FLASHBACK DATABASE TO SCN 5964663
FLASHBACK DATABASE TO BEFORE SCN 5964663
FLASHBACK DATABASE TO TIMESTAMP (SYSDATE -
1/24)
FLASHBACK DATABASE TO SEQUENCE 12345
```

2. Open the database with `READ ONLY` option to check that the database flashed back to the correct time.

3. If you decide to go back further in time, you can flashback the database again.

4. If you determine that you flashed back too far into the past, you can use redo logs to roll forward.

5. Open the database with `RESETLOGS` option:

```
ALTER DATABASE OPEN RESETLOGS
```

6. If you want to completely undo the effects of the flashback database operation, just use the command `RECOVER DATABASE` to perform a complete recovery of the database.

Displaying Flashback Storage Information

In order to estimate the space you need to add to your flash recovery area for accommodating the flashback database logs:

```
SELECT ESTIMATED_FLASHBACK_SIZE,
RETENTION_TARGET, FLASHBACK_SIZE FROM
V$FLASHBACK_DATABASE_LOG
```

To really know how far back you can flashback your database at any given time, you must query the `V$FLASHBACK_DATABASE_LOG` in the following manner:

```
SELECT OLDEST_FLASHBACK_SCN,
OLDEST_FLASHBACK_TIME FROM
V$FLASHBACK_DATABASE_LOG
```

The view `V$FLASHBACK_DATABASE_STATS` helps you monitor the I/O overhead of logging flashback data.

`BEGIN_TIME` and `END_TIME` stand for the beginning and ending hourly time intervals for which the view's statistics were collected. Oracle collects flashback data on an hourly basis for a maximum of 24 hours. If you issue a query on the table, however, it may return 25 rows, the 25th row being for the most recent fraction of time after the last (24th) row was logged in the view.

`FLASHBACK_DATA` stands for the number of bytes of flashback data written during the interval.

`DB_DATA` stands for the number of bytes of database data read and written during the interval.

`REDO_DATA` stands for the number of bytes of redo data written during the interval.

`ESTIMATED_FLASHBACK_SIZE` is identical to the value of the `ESTIMATED_FLASHBACK_SIZE` column in the `V$FLASHBACK_DATABASE_LOG` view.

Flashback Drop

How the Flashback Drop Feature Works

- When you issue the `DROP TABLE` command, Oracle merely renames the table and moves it to a recycle bin.
- The recycle bin is merely a data dictionary table that maintains information about dropped tables.
- You can use the `SELECT` command to query the objects in the recycle bin. You can't use `INSERT`, `UPDATE`, and `DELETE` commands on these objects.

Querying the Recycle Bin

You can view the contents of the recycle bin by using either the `DBA_RECYCLEBIN` or `USER_RECYCLEBIN`.

Alternatively, you can use the `SHOW RECYCLEBIN` command which shows only those objects that you can undrop.

Restoring Dropped Tables

In order to restore a dropped table:

```
FLASHBACK TABLE persons TO BEFORE DROP
FLASHBACK TABLE
"BIN$ksisyg0TxKnt18rqukpQA==$0"
TO BEFORE DROP RENAME TO NEW_PERSONS
```

Note: When you flashback a table, Oracle will recover the dependent objects as well, but they'll continue to have their cryptic system-generated names.

If you drop and re-create a table with the same name, the recycle bin will have several versions of the dropped table, each with a unique system-generated table name. If you then issue a `FLASHBACK TABLE... TO BEFORE DROP` command, Oracle will simply recover the latest version of the table. If you don't want Oracle to do this, you have the following options:

- In the `FLASHBACK TABLE` command, provide the specific system-generated name of the table you want to recover.
- Keep issuing the `FLASHBACK TABLE` command until you recover the particular table you want.

Permanently Removing Tables

```
DROP TABLE PERSONS PURGE
PURGE TABLE "BIN$Q1QZGCCMRSSCBBRN9IVWFA==$0"
PURGE TABLESPACE USERS USER SCOTT
```

`PURGE RECYCLEBIN` or `PURGE USER_RECYCLEBIN` will remove all objects belonging to the user issuing the command.

`PURGE DBA_RECYCLEBIN` command will remove all objects in the recycle bin. You must have the `SYSDBA` privilege to purge the entire recycle bin.

If you drop a tablespace, any objects belonging to the tablespace that are part of the recycle bin are purged immediately.

If you use the command `DROP USER ... CASCADE`, any objects in the recycle bin that belong to that user are automatically purged.

Restrictions on Flashback Drop

- Table should belong to any non-SYSTEM, locally managed tablespace.
- Dependent objects can be in either a locally or dictionary managed tablespace, to be stored in the recycle bin.
- The following types of dependent objects aren't saved in the recycle bin:
 - Materialized view logs
 - Referential integrity constraints
 - Bitmap join indexes
- You can't save a table that has fine-grained auditing (FGA) or Virtual Private Database policies defined on it.

Flashback Table

How Flashback Table Works

Flashback table technology uses undo information to restore data rows in changed blocks of tables.

Pre-requisites

- You must have either the `FLASHBACK ANY TABLE` or the more specific `FLASHBACK` object privilege on the table you want to recover. In addition, you must have the `SELECT`, `INSERT`, `DELETE`, and `ALTER` privileges on the table.
- Make sure you enable row movement in the table:

```
ALTER TABLE persons ENABLE ROW MOVEMENT
```

How to Flashback a Table

First, it is useful to note the current SCN then issue the command:

```
FLASHBACK TABLE persons TO SCN 6039341
```

```
FLASHBACK TABLE persons TO TIMESTAMP
TO_TIMESTAMP('2004-07-04 08:05:00', 'YYYY-MM-DD
HH24:MI:SS')
```

Oracle disables all relevant triggers by default and reenables them upon completing the table recovery. You may simply append the `ENABLE TRIGGERS` clause to your `FLASHBACK TABLE` command if you want to override this default behavior.

The `persons` table continues to be online and accessible to users for all queries. However, Oracle acquires exclusive DML locks on the table during the Flashback Table operation.

Undoing a Flashback Table Operation

It is important to note your current SCN before using a Flashback Table operation.

Use the `FLASHBACK TABLE` statement again to go back to just before you were when you issued the first statement.

Restrictions on Flashback Table

- You can't flashback a system or remote table.
- You can't flashback a table back to a time preceding any DDL operation that changes the structure of a table (for example, adding or dropping a column).
- Oracle doesn't flashback statistics of the recovered objects.

Row Level Flashback Features

The value of the `UNDO_RETENTION` parameter determines the length of time your users can flashback their queries.

Flashback Query (`SELECT...AS OF`)

```
SELECT * FROM persons AS OF TIMESTAMP
TO_TIMESTAMP('2004-07-04 08:05:00', 'YYYY-MM-DD
HH:MI:SS') WHERE NAME = 'ALAPATI'
```

Flashback Versions Query

When you issue a `SELECT` statement using the `VERSIONS` clause, Oracle will return the different committed versions of the same row between two SCNs or two timestamps.

```
VERSIONS BETWEEN
{SCN | TIMESTAMP} start|MINVALUE AND
end|MAXVALUE
```

```
[AS OF {SCN|TIMESTAMP expr}]
```

Here is a brief explanation of pseudocolumns that will be part of the flashback versions query output:

VERSIONS_STARTSCN and VERSIONS_STARTTIME This pseudocolumn tells you the SCN and timestamp when this particular row was first created.

VERSIONS_ENDSCN and VERSIONS_ENDTIME These pseudocolumns tell you when this particular row expired.

VERSIONS_OPERATION This pseudocolumn provides you with information as to the type of DML activity that was performed on the particular row. The DML activities are indicated by letters: **I** stands for insert, **D** for delete, and **U** for update.

VERSIONS_XID This pseudocolumn stands for the unique transaction identifier of the transaction that resulted in this row version.

Note: If the **VERSIONS_STARTSCN** and the **VERSIONS_STARTTIME** are **NULL**, then the row was created before the lower bound specified by your **BETWEEN** clause.

Note: If the **VERSIONS_ENDSCN** and the **VERSIONS_ENDTIME** are **NULL**, this means that this row version is current when you tried to use the Flashback Versions Query operation, or the row was part of a delete operation.

Note: An index-organized table (IOT) will show an update operation as a separate insert and a delete operation.

Example:

```
SELECT VERSIONS_XID XID, VERSIONS_STARTSCN
START_SCN, VERSIONS_ENDSCN END_SCN,
VERSIONS_OPERATION OPERATION, empname, salary
FROM hr.emp
VERSIONS BETWEEN SCN MINVALUE AND MAXVALUE
AS OF SCN 113900
WHERE empno = 111
```

Flashback Transaction Query

FLASHBACK_TRANSACTION_QUERY lets you identify which transaction or transactions were responsible for certain changes during a certain interval.

Its columns are:

```
XID, START_SCN, START_TIMESTAMP, COMMIT_SCN,
COMMIT_TIMESTAMP, LOGON_USER, UNDO_CHANGE#,
OPERATION, TABLE_NAME, TABLE_OWNER, ROW_ID,
UNDO_SQL
```

Note: You must have the **SELECT ANY TRANSACTION** system privilege to query the **FLASHBACK_TRANSACTION_QUERY** view.

Using Flashback Transaction Query and Flashback Versions Query

```
SELECT XID, START_SCN START, COMMIT_SCN COMMIT,
OPERATION OP, LOGON_USER USER,
UNDO_SQL FROM FLASHBACK_TRANSACTION_QUERY
WHERE XID = HEXTORAW('000200030000002D')
```

Value passed to **HEXTORAW** function obtained from Flashback versions query of an old row version to undo or audit.

Flashback Transaction Query Considerations

- Flashback Transaction Query on a transaction underlying a DDL displays the changes made to the data dictionary.
- When you use Flashback Transaction Query on a dropped table, object number (not the table name) will be displayed.
- When you use Flashback Transaction Query on a dropped table, userid (not the username) will be displayed.
- If you query a transaction involving an IOT, an update operation is always shown as a two-step delete/insert operation.
- Sometimes you'll notice a value of **UNKNOWN** under the **OPERATION** column, if the transaction didn't have enough undo information to correctly identify its operation type.
- You may want to turn on minimal supplemental logging in order to support operations involving chained rows and special storage structures such as clustered tables.

```
ALTER DATABASE ADD SUPPLEMENT LOG DATA
```

Automatic Storage Management

Introduction to Automatic Storage Management

ASM acts as Oracle's own Logical Volume Manager (LVM), by handling striping and mirroring functions previously done by third party tools.

You can't use operating system commands or utilities to access ASM files. You must use the **RMAN** to copy ASM files.

ASM Instance Architecture

- ASM has three important components: the ASM instance, disk groups, and ASM files.
- An ASM instance has several background processes like the **SMON**, **PMON**, and **LGWR** processes. In addition, there are two new background processes: **ASM Rebalance Master (RBAL)** and **ASM Rebalance (ARBn)**.
- Any Oracle database instance that uses an ASM instance will have two new ASM-related background processes, the **RBAL** and the **ASM Background (ASMB)** processes.
- ASM Files backup must be made by **RMAN**

Managing the ASM Instance

Initialization Parameters for the ASM Instance

INSTANCE_TYPE

You must set the **INSTANCE_TYPE** parameter to **ASM**.

DB_UNIQUE_NAME

This parameter applies only to ASM within a cluster or on a node. The parameter shows the unique name for

a group of ASM instances in a cluster or on a node. The default value for this parameter is +ASM.

ASM_POWER_LIMIT

This parameter indicates the maximum speed to be used by this ASM instance during a disk rebalance operation. The default for this parameter is 1, and the range is 1 (slowest) to 11 (fastest).

ASM_DISKSTRING

This parameter sets the disk location for Oracle to consider during a disk-discovery process. Default is NULL which means ASM will find all disks to which it has read/write access.

```
ASM_DISKSTRING = '/dev/rdisk/*s1',
'/dev/rdisk/c1*
```

ASM_DISKGROUPS

This parameter lets you specify the name of any disk group that you want the ASM instance to automatically mount at instance startup. The default value for this parameter is NULL.

If you use an init.ora text file, you must make sure to add the names of any disk groups that you want to mount when the instance starts up. If you use an SPFILE, Oracle will automatically make the necessary additions and deletions to the SPFILE when you create, add, or drop a disk group.

Note: The ASM instance uses the `LARGE_POOL` memory buffer. You should allocate at least 8MB to this parameter, so it can serve the ASM instance effectively. Most ASM instances should need no more than 64MB of SGA.

Note: If you set only one parameter `INSTANCE_TYPE=ASM`, Oracle will start up the ASM instance with default values for all the other parameters.

Creating the ASM Instance using DBCA

While you use DBCA to create a database, if you choose ASM for storage, the DBCA will check to see if an ASM instance already exists on your server. If it does, the DBCA will then show you the disk groups being managed by that ASM instance and ask you to choose the disk groups for your new Oracle database. If you haven't already configured an ASM instance, the DBCA will automatically create one for you.

The DBCA automatically creates an entry in the `oratab` file on UNIX systems, so the operating system is aware of the new instance. On Windows systems, the DBCA creates the Oracle service and makes the appropriate Windows Registry entries. The DBCA also creates a parameter file (spfile) and a password file for the new ASM instance.

Creating the ASM Instance Manually (on Windows)

1. Building the ASM Candidate "disks": for testing or development purpose

```
ASMT00L -create c:\asmdisks\asmdisk1 250
```

2. Create a pfile with the name "init+ASM.ora" in the folder `<ORACLE_HOME>\database`. Insert the following parameters in the file:

```
INSTANCE_TYPE=ASM
_ASM_ALLOW_ONLY_RAW_DISKS = FALSE
DB_UNIQUE_NAME = +ASM
ASM_DISKSTRING = 'C:\asmdisks\*'
LARGE_POOL_SIZE = 16M
BACKGROUND_DUMP_DEST =
'D:\oracle\admin\+ASM\bdump'
USER_DUMP_DEST = 'D:\oracle\admin\+ASM\udump'
CORE_DUMP_DEST = 'D:\oracle\admin\+ASM\cdump'
```

3. Create the ASM instance service:

```
ORADIM -NEW -ASMSID +ASM -STARTMODE auto
```

4. Startup the instance

```
SET ORACLE_SID=+ASM
C:\> SQLPLUS / AS SYSDBA
SQL> STARTUP FORCE
SQL> SELECT PATH, MOUNT_STATUS FROM V$ASM_DISK;
```

Creating the ASM Instance Manually (on Unix)

Steps here assumes the following:

- Red Hat Enterprise Server 3 installed and patched to kernel version 2.4.21-15
- Oracle version 10.1.0.3 (Enterprise Edition) installed as per instructions here.

1. After logging as `root`, create disks

Create physical files:

```
dd if=/dev/zero of=/asmdisks/disk1 bs=1024k
count=250
```

Map loopback devices to the files:

```
/sbin/losetup /dev/loop1 /asmdisks/disk1
```

2. Download `oracleasm` utility from Oracle site.

3. Install the utility files as follows:

```
rpm -ivh *.rpm
```

4. With the basic libraries installed, you need to configure them so that they get re-loaded at every server reboot:

```
[root@koala howardjr]# /etc/init.d/oracleasm
configure
```

```
Default user to own the driver interface []:
oracle
```

```
Default group to own the driver interface []:
oinstall
```

```
Start Oracle ASM library driver on boot (y/n)
[n]: y
```

```
Fix permissions of Oracle ASM disks on boot
(y/n) [y]: y
```

5. Writing the ASM Disk Header information:

```
/etc/init.d/oracleasm createdisk ASMD1
/dev/loop1
```

```
Marking disk "/dev/loop1" as an ASM disk [ OK ]
```

6. After logging on as `Oracle` user now, under the `$ORACLE_HOME/dbs` directory, create the file "init+ASM.ora" and type the following in it:

```
INSTANCE_TYPE = ASM
DB_UNIQUE_NAME = +ASM
LARGE_POOL_SIZE = 16M
ASM_DISKSTRING = 'ORCL:*
```

```
[oracle@koala dbs]$ export ORACLE_SID=+ASM
[oracle@koala dbs]$ sqlplus / as sysdba
SQL> startup
SQL> select path from v$asm_disk;
```

Starting and Shutting Down an ASM Instance

- When starting an ASM instance, you can use the `STARTUP` command with the `NOMOUNT`, `MOUNT`, `RESTRICT` and `FORCE` options. You cannot use the `STARTUP OPEN` syntax.
- If you either start up your ASM instance with the `STARTUP RESTRICT` command or issue the `ALTER SYSTEM ENABLE RESTRICTED SESSION` command in a

normal ASM instance, Oracle database instances cannot connect to the ASM instance.

- If you shut down an ASM instance, all Oracle databases currently connected to it will also shut down.

Managing ASM Disk Groups

ASM Striping

- For performance reasons, you must use disks of the same type and performance capacity in a disk group.
- ASM provides two types of data striping, depending on the database file type:

Coarse striping: The stripe size is a relatively large 1MB chunk of file space. You may use coarse striping for all files in an Oracle database, except the control files, online redo log files, and flashback files.

Fine striping To reduce file latency, ASM provides a fine striping scheme, where the striping is in smaller chunk sizes of 128KB. You may want to use fine striping for control files, online redo log files, and flashback files.

ASM Mirroring

Disk mirroring provides data redundancy. If you lose a disk, you can use its mirror disk to continue operations without missing a beat. ASM mirrors extents.

Failure Groups

Failure groups define disks that share components, such that if one fails then other disks sharing the component might also fail.

Types of ASM Mirroring

- **External redundancy** You choose this level of mirroring when you are using operating system storage array protection. Disk groups under this redundancy level don't have any failure groups.
- **Normal redundancy** This type provides two-way mirroring. Thus, to support a normal redundancy level, you must create at least *two* failure groups.
- **High redundancy** This type provides three-way mirroring. You must create at least *three* failure groups.

Creating a Disk Group

```
SQL> STARTUP NOMOUNT
SQL> CREATE DISKGROUP dgroup1 NORMAL REDUNDANCY
  FAILGROUP controller1 DISK
    '/devices/diska1' name testdisk size 100G,
    '/devices/diska2',
    '/devices/diska3'
  FAILGROUP controller2 DISK
    '/devices/diskb1',
    '/devices/diskb2',
    '/devices/diskb3'
```

You can force a disk that is already a member of another disk group to become a member of the disk group you are creating by specifying the `FORCE`

Note: The `CREATE DISKGROUP` statement mounts the disk group for the first time, and adds the disk group name to the `ASM_DISKGROUPS` initialization parameter if a spfile is being used. If a pfile is being used and you want the disk group to be automatically mounted at

instance startup, then you must add the disk group name to the `ASM_DISKGROUPS` initialization parameter before the next time that you shut down and restart the ASM instance.

Adding Disks to a Disk Group

```
ALTER DISKGROUP dgroup1 ADD DISK
  '/devices/diska5' NAME diska5,
  '/devices/diska6' NAME diska6;
ALTER DISKGROUP dgroup1 ADD DISK
  '/devices/diska*';
```

- When a disk is added, it is formatted and then rebalanced.
- When you don't specify a `FAILGROUP` clause, the disk is in its own failure group.
- If you don't specify the `NAME` clause, Oracle assigns its own system-generated names.
- If the disk already belongs to a disk group, the statement will fail.
- Use the `FORCE` clause to add a disk that is a current member of disk group.

Dropping Disks and Disk Groups

```
ALTER DISKGROUP dgroup1 DROP DISK diska5;
DROP DISKGROUP test_groupa INCLUDING CONTENTS;
```

- `DROPT DISKGROUP` statements requires the instance to be in `MOUNT` state.
- When a disk is dropped, the disk group is rebalanced by moving all of the file extents from the dropped disk to other disks in the disk group. The header on the dropped disk is then cleared.
- If you specify the `FORCE` clause for the drop operation, the disk is dropped even if Automatic Storage Management cannot read or write to the disk.
- You can also drop all of the disks in specified failure groups using the `DROP DISKS IN FAILGROUP` clause.

Undropping Disks in Disk Groups

```
ALTER DISKGROUP dgroup1 UNDROP DISKS;
```

- This statement enables you to cancel all pending drops of disks within disk groups.

Rebalancing Disk Groups

You can increase the speed of a rebalancing operation by doing any of the following things:

- raising the value of the `ASM_POWER_LIMIT` initialization parameter
- using a high value for the `POWER` clause in a disk rebalance operation

```
ALTER DISKGROUP dgroup1 REBALANCE POWER 5
```
- performing all your disk adding, resizing, and dropping operations at the same time.

Managing ASM Files

Types of ASM Filenames

1. Fully Qualified ASM Filenames (System Alias)

You use this fully qualified name for *referencing* existing ASM files. Here's the syntax of an ASM file using a fully qualified filename:

```
+group/dbname/file_type/tag.file.incarnation
```

2. Numeric ASM Filenames

ASM derives *numeric filenames* from fully qualified ASM filenames and uses them to refer to existing files.

```
+group.file.incarnation
```

3. Alias ASM Filenames

You can use ASM alias files both when *creating* new ASM files and when *referring* to existing files. Alias ASM filenames mean that the files are not OMF-managed files. Thus, Oracle won't automatically remove these files when it does not have any further need for them.

```
+dgroup1/myfiles/control_file1
+dgroup2/mydir/second.dbf
```

4. Incomplete ASM Filenames

You can use an *incomplete ASM filename* only when *creating* files.

```
+dgroup1
+dgroup1 (datafile)
```

Alias Filename Management

Creating Disk Group Directories for Alias Filenames

You must create a *directory structure* to support your alias naming conventions.

```
ALTER DISKGROUP dgroup1 ADD DIRECTORY
'+dgroup1/mydir';
```

Using Templates with Aliases

```
dgroup(template_name)/alias
+dgroup1(spfile)/config1
```

Adding Aliases

You can add a filename alias or rename an existing alias name, using the `ADD ALIAS` or `RENAME ALIAS` clause of the `ALTER DISKGROUP` statement.

```
ALTER DISKGROUP dgroup1 ADD ALIAS
'+dgroup1/mydir/second.dbf' FOR
'+dgroupA/sample/datafile/mytable.342.3'
```

You can retrieve created aliases using `v$ASM_ALIASES`. The `REFERENCE_INDEX` column is usable only for entries that are directory entries in the alias directory. For non-directory entries, it equals to zero.

Dropping Files and Aliases from a Disk Group

```
ALTER DISKGROUP dgroup1 DROP FILE
'+dgroup1/payroll/compensation.dbf'
```

ASM File Templates

Whenever you create a disk group, Oracle establishes a set of initial system default templates for that disk group.

Template Name	External Redundancy	Normal Redundancy	High Redundancy	Striped
CONTROL	Unprotected	2-way mirror	3-way mirror	Fine
DATAFILE	Unprotected	2-way mirror	3-way mirror	Coarse
ONLINELOG	Unprotected	2-way mirror	3-way mirror	Fine
ARCHIVELOG	Unprotected	2-way mirror	3-way mirror	Coarse

You can create your own template:

```
alter diskgroup test_group1 add template
production attributes (mirror fine)
```

You cannot change a file's attributes once you create it using a certain template. If you wish to change an ASM file's attributes, you must use the RMAN to copy the file into a new file with the attributes you want.

Migrating a Database to ASM

Setting Instance Parameters

`INSTANCE_TYPE`: defaults to `RDBMS`

`LOG_ARCHIVE_FORMAT` If you set the `LOG_ARCHIVE_FORMAT` to an incomplete ASM filename (such as `+dgroupA`), Oracle will ignore it. If you set it to an ASM directory, Oracle will use the directory and create non-OMF files in that directory.

You must use incomplete ASM filenames as the destination for the following initialization parameters:

```
DB_CREATE_FILE_DEST_n
DB_CREATE_FILE_DEST
DB_RECOVERY_FILE_DEST
CONTROL_FILES
LOG_ARCHIVE_DEST_n
LOG_ARCHIVE_DEST
STANDBY_ARCHIVE_DEST
```

Creating an ASM-Based Database

You can create an ASM-based database simply by setting the following parameters:

```
DB_CREATE_FILE_DEST = '+dgroup1'
DB_RECOVERY_FILE_DEST = '+dgroup2'
DB_RECOVERY_FILE_DEST_SIZE = 100G
```

Now, commands that require file specifications can be issued easier than before:

```
CREATE DATABASE test
CREATE TABLESPACE test_tbsp
ALTER DATABASE ADD logfile
```

Migrating Your Database to ASM

1. Obtain current control file and redo log files locations using `v$CONTROLFILE` and `v$LOGFILE`
2. Shut down cleanly the database
3. Set the parameters to make the database OMF-based.
4. Delete the control file parameter from your `SPFILE`.
5. Startup the database in `NOMOUNT`
6. Using RMAN issue the following script:

```
RESTORE CONTROLFILE FROM '/u1/c1.ct1';
ALTER DATABASE MOUNT;
BACKUP AS COPY DATABASE FORMAT '+dgroup1';
SWITCH DATABASE TO COPY;
SQL "ALTER DATABASE RENAME '/u1/log1' TO
'+dgroup1' ";
# Repeat RENAME command for all online redo
log members ...
ALTER DATABASE OPEN RESETLOGS;
SQL "alter tablespace temp add tempfile"
SQL "ALTER DATABASE TEMPFILE '/u1/temp1'
DROP";
```

Monitoring Long-Running Operations

The `ALTER DISKGROUP DROP`, `RESIZE`, and `REBALANCE` commands return before the operation is complete. To monitor progress of these long-running operations, you can query the `v$ASM_OPERATION` fixed view.

GROUP_NUMBER Disk group

OPERATION Type of operation: REBAL

STATE State of operation: QUEUED or RUNNING

POWER Power requested for this operation

ACTUAL Power allocated to this operation

SOFAR Number of allocation units moved so far

EST_WORK Estimated number of remaining allocation units

EST_RATE Estimated number of allocation units moved per minute

EST_MINUTES Estimated amount of time (in minutes) for operation termination

e.hire_date, e.job_id, e.salary,
e.commission_pct, e.manager_id,
e.department_id)

In Oracle 10g, you can use a WHERE clause in a MERGE statement's UPDATE or INSERT clause:

```
MERGE USING product_Changes s
INTO products p
ON (p.prod_id = s.prod_id)
WHEN MATCHED THEN UPDATE
SET p.prod_list_price = s.prod_new_price
WHERE p.prod_status <> "EXPIRED"
WHEN NOT MATCHED THEN INSERT
SET p.prod_list_price = s.prod_new_price
WHERE s.prod_status <> "EXPIRED"
```

You can use DELETE clause with MERGE statement and it must be embedded inside the UPDATE statement.

The DELETE clause in a MERGE operation will evaluate only the updated values (values updated by the UPDATE clause) and not the original values that were evaluated by the UPDATE clause.

```
MERGE USING product_changes s
INTO products p ON (d.prod_id = s.prod_id)
WHEN MATCHED THEN
UPDATE SET d.prod_list_price =
s.prod_new_price,
d.prod_status = s.prod_new_status
DELETE WHERE (d.prod_status = "OLD_ITEM")
WHEN NOT MATCHED THEN
INSERT (prod_id, prod_list_price, prod_status)
VALUES (s.prod_id, s.prod_new_price,
s.prod_new_status)
```

Dynamic Performance Views

V\$ASM_DISKGROUP

In an ASM instance, this view provides information about a disk group. In a database instance, this view contains one row for every ASM disk group mounted by the ASM instance.

V\$ASM_CLIENT

In an ASM instance, this view identifies all the client databases using various disk groups. In a Database instance, the view contains one row for the ASM instance if the database has any open ASM files.

V\$ASM_DISK

In an ASM instance, this view contains one row for every disk discovered by the ASM instance. In a database instance, the view will only contain rows for disks in use by that database instance.

V\$ASM_FILE

This view contains one row for every ASM file in every disk group mounted by the ASM instance.

V\$ASM_TEMPLATE

This view contains one row for every template present in every disk group mounted by the ASM instance.

Enhancements in Analytical SQL and Materialized Views

Enhancements in the MERGE Statement

The basic MERGE statement has the following structure:

```
MERGE <hint> INTO <table_name>
USING <table_view_or_query>
ON (<condition>)
When MATCHED THEN <update_clause>
When NOT MATCHED THEN <insert_clause>
```

Example

```
MERGE INTO copy_emp c
USING employees e
ON (c.employee_id = e.employee_id)
WHEN MATCHED THEN
UPDATE SET
c.first_name = e.first_name,
c.last_name = e.last_name,
...
WHEN NOT MATCHED THEN
INSERT VALUES(e.employee_id, e.first_name,
e.last_name, e.email, e.phone_number,
```

Using Partitioned Outer Joins

Partitioned outer joins help turn *sparse* data into *dense* data, you thus have faster performance and a better reporting format.

The partitioned outer join is ideal for time dimensions, but it can be used for any kind of dimensions.

```
SELECT .....
FROM table_reference
PARTITION BY (expr [, expr ]... )
RIGHT OUTER JOIN table_reference
```

and

```
SELECT .....
FROM table_reference
LEFT OUTER JOIN table_reference
PARTITION BY {expr [,expr ]...}
```

Using the SQL MODEL Clause

MODEL clause enables you to generate multidimensional output query in the database.

Example:

```
SELECT country, product, year, sales
FROM sales_view
WHERE country IN ('Mexico', 'Canada')
MODEL
PARTITION BY (country)
DIMENSION BY (product, year)
MEASURES (sale sales)
RULES
(sales['Kleenex', 2005] =
sales['Kleenex', 2004] +
```

```

        sales['Kleenex',2003],
sales['Pampers', 2005] =
        sales['Pampers', 2004],
sales['All_Products', 2005] =
        sales['Kleenex', 2005] +
        sales['Pampers',2005])
ORDER BY country, product, year

```

You can specify that Oracle should evaluate the rules in either of the following two ways:

SEQUENTIAL ORDER: Oracle will evaluate a rule in the order it appears in the `MODEL` clause.

AUTOMATIC ORDER: Oracle will evaluate the rule on the basis of the dependencies between the various rules in the `MODEL` clause.

By default, the `RULES` keyword operates with the `UPSERT` specification. You can use the `UPDATE` option. This specification can be applied in the `RULES` level or in a specific rule-level.

Materialized View Enhancements

- In Oracle Database 10g, the `ENABLE_QUERY_REWRITE` parameter is `TRUE` by default. You must, however, ensure that the `OPTIMIZER_FEATURES_ENABLE` initialization parameter is set to 10.0.0 or higher.
- The `QUERY_REWRITE_INTEGRITY` initialization parameter still has the same default value (`ENFORCED`).
- You can use the following two procedures in `DBMS_MVIEW`:
 - `EXPLAIN_MVIEW` This procedure tells you what kinds of query rewrites are possible. It will also tell you why a certain materialized view is not fast refreshable.
 - `EXPLAIN_REWRITE` This procedure tells you *why* a query failed to rewrite. If the query rewrites, the procedure will tell you which materialized views will be used.

Using the `DBMS_ADVISOR.TUNE_MVIEW` procedure

The `DBMS_ADVISOR.TUNE_MVIEW` procedure recommends materialized views with optimized defining queries, decomposition of nonrefreshable materialized views, and fixes for materialized view log problems. It also tells you how to make a materialized view eligible for a fast refresh, if it is not.

```

begin
DBMS_ADVISOR.TUNE_MVIEW (:task_name,
'CREATE MATERIALIZED VIEW test_mv
REFRESH FAST WITH ROWID ENABLE QUERY REWRITE
AS SELECT DISTINCT prod_name, prod_type
From products');
end;

```

The preceding code will populate the new `DBA_TUNE_MVIEW` view.

<code>TASK_NAME</code>	to identify and query a particular <code>TUNE_MVIEW</code> recommendation.
<code>ACTION_ID</code>	column shows the command order number.
<code>SCRIPT_TYPE</code>	<code>CREATE</code> , <code>DROP</code> , <code>UNKNOWN</code>
<code>STATEMENT</code>	shows the recommended materialized view changes that

make your materialized view eligible for a fast refresh and a query rewrite.

```

SELECT STATEMENT
FROM DBA_TUNE_MVIEW
WHERE TASK_NAME = :task_name
ORDER BY SCRIPT_TYPE, ACTION_ID

```

You can use the `DBMS_ADVISOR.GET_TASK_SCRIPT` procedure to output the recommendations to a text file.

Creating Materialized View Logs

one of the restrictions on the fast refresh feature is that you must include the `ROWIDs` of all tables that are in the `FROM` list in your `SELECT` list.

```

CREATE MATERIALIZED VIEW LOG ONEMPLOYEES
WITH SEQUENCE, ROWID INCLUDING NEW VALUES

```

Decomposing Materialized Views

`TUNE_MVIEW` procedure may make recommendations for the decomposition of the materialized view into two nested submaterialized views. The parent materialized view will refer to the submaterialized view that you create. This occurs in the following situations:

- A subquery in the `WHERE` clause
- Use of set operations like `UNION`, `UNION ALL`, `INTERSECT`, and `MINUS`
- Use of inline views

Partition Change Tracking Enhancements

Any time you change a base table's partition scheme, the relevant materialized view rows become stale. Oracle's partition change tracking (PCT) feature lets you figure out which rows of a materialized view are affected by a change in a base table's partitioning.

Oracle Database 10g extends the use of PCT to list-partitioned tables, enables the use of `ROWID` columns as partition markers, and lets you use a PCT refresh if a materialized view contains a join-dependent expression.

The `DBMS_MVIEW.REFRESH` procedure has a new option, `P`, to indicate a forced PCT-based refresh:

```

DBMS_MVIEW.REFRESH(mview_name, method =>'P')

```

Materialized View Execution Plans

The explain plan feature shows you whether a materialized view is being accessed as a result of a query rewrite or because you specified direct materialized view access.

Using the `V$SQL_PLAN` view:

```

Query Plan
SELECT STATEMENT
SORT ORDER BY
MATERIALIZED VIEW REWRITE ACCESS FULL EMP_INFO

```

If you don't see the keyword `REWRITE`, it means that the materialized view was accessed directly.

The `REWRITE_OR_ERROR` Hint

Oracle Database 10g contains a new optimizer hint called `REWRITE_OR_ERROR`, which forces a query to error out if it can't rewrite the query:

```

SELECT /*+ REWRITE_OR_ERROR */ ...

```

ORA-30393: A query block in the statement did not rewrite

New Columns in the `REWRITE_TABLE`

If `REWRITE_OR_ERROR` raised, you can use the `DBMS_MVIEW.EXPLAIN_REWRITE` procedure to find out why the query failed to rewrite.

1. Create the `REWRITE_TABLE` table:

```
<ORACLE_HOME>\RDBMS\ADMIN\utlrxrw.sql

STATEMENT_ID      ID for the query
MV_OWNER          MV's schema
MV_NAME           Name of the MV
SEQUENCEINTEGER  Seq # of error msg
QUERY             user query
MESSAGE           EXPLAIN_REWRITE error msg
PASS              Query Rewrite pass no
MV_IN_MSG         MV in current message
MEASURE_IN_MSG    Measure in current message
JOIN_BACK_TBL     Join back table in current msg
JOIN_BACK_COL     Join back column in current msg
ORIGINAL_COST     Cost of original query
INTEGER
REWRITTEN_COST    Cost of rewritten query. It shows a
                  zero if there was no rewrite of a
                  query or if a different materialized
                  view was used
FLAGS             Associated flags
```

2. Execute `DBMS_MVIEW.EXPLAIN_REWRITE`:

```
DBMS_MVIEW.EXPLAIN_REWRITE ('SELECT
p.prod_name, SUM(amount_sold).. ',
'TestXRW.PRODUCT_SALES_MV', 'SH')

SELECT message FROM rewrite_table ORDER BY
sequence;

MESSAGE
-----
QSM-01033: query rewritten with materialized
view, PRODUCT_SALES_MV
```

Materialized Join View Enhancements

Materialized join views (MJVs) contain only joins (and not aggregates).

For a fast refresh of materialized join views — whether they use self joins, inline views, or remote tables — you must create materialized view logs on each of the base tables. The materialized view logs must also contain the `ROWID` column.

Partition Maintenance Operations

In Oracle Database 10g, you can issue commands that truncate, exchange, or drop partitions by using the `ALTER MATERIALIZED VIEW` statement.

Materialized View Refresh Using Trusted Constraints

If you use the `TRUSTED` option, the resulting materialized views are in an unknown state, and you can use them for a query rewrite in a `TRUSTED` or a `STALE_TOLERATED` mode only.

Miscellaneous New Features

VPD and Auditing Enhancements

VPD policies apply to tables, views, and synonyms. You can apply VPD policies to `SELECT`, `INSERT`, `DELETE`, `UPDATE`, and any `INDEX` statements.

Column-Level VPD

A column-level VPD policy applies only to tables and views and not to synonyms. You may apply a policy function to queries as well as DML statements.

When you use column-level VPD, you have a choice of two types of behavior by the policy:

- **Default behavior** will restrict the number of rows returned by any query that contains the security-relevant columns(s).
- **Column-masking** behavior, on the other hand, will return all the rows, but show null values for the security-relevant columns in those rows.

Creating a Column-Level Policy

```
DBMS_RLS.ADD_POLICY (OBJECT_SCHEMA=>'scott',
                    OBJECT_NAME=>'emp',
                    POLICY_NAME=>'test_policy',
                    FUNCTION_SCHEMA=>'test_schema',
                    POLICY_FUNCTION=>'test_function',
                    STATEMENT_TYPE='insert,update',
                    SEC_RELEVANT_COLS=>'salary,commission')
```

Note: You can implement column-masking behavior by using the `SEC_RELEVANT_COLS_OPT => DBMS_RLS.ALL_ROWS` parameter.

Note: The default of `STATEMENT_TYPE` is to apply to all the types except `INDEX`.

A function policy can be created as in the following:

```
CREATE OR REPLACE FUNCTION test_function
(objowner IN VARCHAR2, objname IN VARCHAR2)
RETURN VARCHAR2 AS
con VARCHAR2(200);
BEGIN
  con := 'deptno = 5';
  RETURN (con);
END test_function;
```

Note: You can grant the privilege `GRANT EXEMPT ACCESS POLICY` to a user so that he or she may bypass a security policy.

New Policy Types

Dynamic

By default, Oracle VPD policy functions are dynamic in nature. That is, Oracle will execute the security policy statement each time a DML statement refers to it and this leads to high resources consumption.

Static Policies

The database executes a static policy function just once, and caches the predicate resulting from the policy evaluation in the SGA. It then applies this predicate to all queries accessing the protected objects.

Static policy can be defined in `DBMS_RLS.ADD_POLICY` by passing the following parameter `POLICY_TYPE => DBMS_RLS.STATIC`

If you want to allow the sharing of the same static policy function over different database objects, you can set the `POLICY_TYPE` parameter to the following value:

```
POLICY_TYPE => DBMS_RLS.SHARED_STATIC
```

Context-Sensitive Policies

These policies will change based on any session context changes.

Context-sensitive VPD policies are particularly useful in a web-based application with connection pooling, where a session needs to change its behavior depending on the `CLIENT_IDENTIFIER` of the user using the session at any given moment.

Context-Sensitive policies can be defined in `DBMS_RLS.ADD_POLICY` by passing the following parameter `POLICY_TYPE => DBMS_RLS.CONTEXT_SENSITIVE`

If you want to allow the sharing of the same context-sensitive policy function over different database objects:

```
POLICY_TYPE => DBMS_RLS.SHARED_CONTEXT_SENSITIVE
```

Auditing Enhancements

Uniform Auditing Trails

Oracle Database 10g helps you audit database activities in a uniform manner by using a new uniform audit trail for both standard and fine-grained audit log records.

`DBMS_FGA` package is used for administering fine-grained audit policies. The `ADD_POLICY` procedure in the package has a parameter `AUDIT_COLUMN_OPTS` which establishes whether a statement is audited when the query references any column specified in the `AUDIT_COLUMN` parameter or only when all such columns are referenced. Possible values are: `ANY_COLUMNS`, `ALL_COLUMNS`.

You can view the new SCN and SQL text/bind variable information only if you use the new `AUDIT_TRAIL=DB_EXTENDED` specification in your initialization parameter file.

Enterprise User Auditing

When you use an LDAP-compliant directory like the Oracle Internet Directory, your users are known as enterprise users. Oracle Database 10g lets you audit the activities of the enterprise users in the database.

Fine-Grained Auditing Enhancements

- You can audit `SELECT`, `INSERT`, `UPDATE`, `DELETE`, and `MERGE` statements.
- You can provide more than one relevant column for fine-grained auditing.
- You can now use `NULL` fine-grained auditing policy predicates.
- Since fine-grained auditing imposes significant SQL information overhead, you can avoid the writing of SQL text and SQL bind information to LOBs.

FGA and DML Statements

- Oracle will audit a DML statement with an FGA policy defined on it if the data rows (old and new) qualify under the policy predicate.
- If you have a relevant column(s) in the security policy, the DML statement will be audited only if it references the column(s) and the data meets the FGA policy requirements.
- Oracle's FGA feature audits `MERGE` statements by viewing the `INSERT` and `DELETE` statements in the `MERGE` statement as individual statements. If there are applicable FGA policies for the `INSERT` or `UPDATE` statement, Oracle will audit the `MERGE` statement.

Enhancements in Managing Multitier Environments

New Dimensions for Statistics Collection and Tracing

The new dimensions for collecting statistics are:

- Client identifier
- Service name
- Combinations of service name, module name, and action name

Enabling Collection of Client and Service Statistics

For client-level Statistics use:

```
DBMS_MONITOR.CLIENT_ID_STAT_ENABLE(<client_id>)
DBMS_MONITOR.CLIENT_ID_STAT_DISABLE(<Client_id>
)
```

For Service-Level Statistics:

```
DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE(<service_name>,<module_name>, <action_name>)
```

Note: The `DBMS_APPLICATION_INFO` has two procedures, `SET_MODULE` and `SET_ACTION`, which allow programmers to specify module and action names.

Viewing the New Statistics

Once you have enabled the collection of the new client identifier, service, module, and action names statistics, you can view them by using Database Control. There are also several new views:

```
DBA_ENABLED_AGGREGATIONS Displays information about enabled statistics aggregation
DBA_ENABLED_TRACES Shows all enabled traces in the system
V$CLIENT_STATS Displays statistics on a client level (CLIENT_IDENTIFIER based)
V$SERVICE_STATS Displays basic performance statistics
V$SERV_MOD_ACT_STATS Displays statistics for a combination of serve /module/action names.
```

Using the TRCSESS Tool to Analyze Trace Files

You can use Oracle's `TRCSESS` command-line utility to consolidate the information from all your trace files into a single output file.

```
trcse ss output=<user.trc> clientid=<user_name> *.trc
```

SQL and PL/SQL Enhancements

UTL_COMPRESS Package

Oracle Database 10g provides the new UTL_COMPRESS package to compress and uncompress data, with the compressed output compatible with the output of the familiar GZIP and GUNZIP compression utilities.

UTL_MAIL Package

In order to use the UTL_MAIL package to send email, you must first execute the utlmail.sql and prvtpmail.plb scripts located in your ORACLE_HOME/rdbms/admin directory.

Regular Expressions

Oracle provides the following regular expression functions for text complex searching:

- o REGEXP_LIKE
- o REGEXP_REPLACE
- o REGEXP_INSTRING
- o REGEXP_SUBSTRING

Case-Insensitive and Accent-Insensitive Query and Sort

When you use the NLS_SORT parameter, you can use the optional suffixes AI or CI to specify whether the sort is accent insensitive (AI) or case insensitive (CI).

```
NLS_SORT = <NLS_sort_name>[_AI|_CI]
NLS_SORT = FRENCH_M_AI
```

CLOB and NCLOB Implicit Conversions

Oracle Database 10g introduces the implicit conversion between LOBs and NCLOBs. Oracle now supports implicit conversion in SQL IN/OUT bind variables, PL/SQL function and procedure parameter passing, and PL/SQL variable assignment.

User-Specified Quoting Characters

You use the new quote operator \grave{a} to provide your own quotation mark delimiters.

Miscellaneous Enhancements

Easy Connect Naming Method

The only condition for using the easy connect naming method is that you should have support for the TCP/IP protocol on both the client and the server.

The new easy connect method is referred to as EZCONNECT in a sqlnet.ora file.

```
Connect
username/password@[//]host[:port][/]service_name
]
```

Only the host name is mandatory.

Simplified Shared Server Configuration

A dispatcher will start automatically when you start a database instance, but no shared server process will start. If you want to start a shared server while your instance is running, you can do so by setting a non-zero value for the SHARED_SERVER initialization parameter, as shown here:

```
ALTER SYSTEM SET SHARED_SERVERS=4
```

Enabling Resumable Space Allocation

RESUMABLE_TIMEOUT parameter enables resumable statements at the system or the session level in seconds. Its default is zero which means it is disabled. In the session level, the following statement should be issued as well:

```
Execute the ALTER SESSION ENABLE RESUMABLE
statement.
```

Flushing the Buffer Cache

```
ALTER SYSTEM FLUSH BUFFER CACHE
```

LogMiner Enhancements

Automatic Adding of Redo Log Files

You can now simply specify a time or SCN, and LogMiner will automatically add the necessary redo log files by scanning the control files for the log information. You must use the DBMS_LOGMNR.CONTINUOUS_MINE procedure to facilitate this automatic gathering of redo log files for mining purposes.

Disabling Generation of ROWIDs

You can disable the generation of physical ROWIDs by using the NO_ROWID_IN_STMT option when you use the DBMS_LOGMNR package.

Easier Removal of Redo Log Files

To remove redo log files, you can now use the new REMOVE_LOGFILE procedure with the DBMS_LOGMNR package.

Automatic Checkpoint Tuning

In Oracle Database 10g, there is no need for you to set the FAST_START_MTTR_TARGET parameter because Oracle itself will automatically tune the checkpointing process.

You can enable automatic checkpoint tuning by simply setting the FAST_START_MTTR_TARGET parameter to any non-zero value.

Copyright

Anyone has the right to copy this document to any means of storage and present it in any format to any individual or organization for free.

If any one wishes to correct a statement or a typing error or add a new piece of information, please send the request to ahmed_b72@yahoo.com. If the modification is acceptable, it will be added to the document, the version of the document will be incremented and the modifier name will be listed in the version history list.

Version History

Version	Individual Name	Date	Updates
1.0	Ahmed Baraka	Sept, 2005	Initial document.