Oracle 10g New Features for Administrators (*Summary Sheets*) v. 2.0

Including Release 2 Features

Version History

Version	Individual Name	Date	Updates
1.0	Ahmed Baraka	Sept, 2005	Initial document.
2.0	Ahmed Baraka	May, 2007	Release 2 features included

Installation, Server Configuration, and Database Upgrades

Comparison Between 10.1 and 10.2

Version	10.1	10.2
Supported Parameters	255	258
Unsupported Parameters	918	1127
Dynamic Performance Views (V$)	340	396
Fixed Views (X$)	529	597
Events (Waits)	811	874
Statistics	332	363
Latches	348	382
Background Processes (Fixed SGA)	109	157

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 `COMPATIBILE` 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.

Managing Database Control

Important EM Agent Directories

When you install Oracle Database 10g, a set of directories and files related to Enterprise Manager is created in the Oracle Home directory:

- `emca` and `emctl` utilities are installed in the `ORACLE_HOME/bin`
- Files that are shared among all instances of the database are stored in `ORACLE_HOME/sysman`
- Files that are unique to each instance of the database are stored in `ORACLE_HOME/hostname_sid/`
- The log files for the Management Agent for that instance are installed in `ORACLE_HOME/hostname_sid/sysman/log/`
- The files required to deploy the Database Control application are installed in the `ORACLE_HOME/oc4j/j2ee` directory structure.
- The `emd.properties` and `emoms.properties` files store agent run-time parameters, and `targets.xml` lists the configured targets.

Configuring Database Control

You can use the operating system command line to configure Database Control. You can use Enterprise Manager Configuration Assistant (EMCA) to perform the following tasks:

- specify the automatic daily backup options.

  ```
  emca -backup
  ```

- add or remove the Enterprise Manager configuration, including the management repository.

  ```
  emca -config dbcontrol db [-repos
  create|recreate]
  emca -deconfig dbcontrol db [-repos drop]
  ```

- reconfigure the default ports used by Enterprise Manager

  ```
  emca -reconfig ports -DBCONTROL_HTTP_PORT
  5500
  ```

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 (in release 2 it is named as **Database Configuration**). 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 **`utlu10*i.sql`.** The DBCA automatically runs it as part of the pre-upgrade check.

Note: In Oracle 10g Release 2, the Pre-Upgrade Information Utility (`utlu102i.sql`) has been enhanced to provide improved resource estimations for tablespace space usage and elapsed upgrade runtime.

The Post-Upgrade Status Tool

Oracle Database 10g also provides a Post-Upgrade Status Tool (**`utlu10*s.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:

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

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

5. If you are upgrading a cluster database and your `initdb_name.ora` file resides within the old environment's Oracle home, then move or copy the `initdb_name.ora` file to the new Oracle home. Make modifications in the file in the same way as made in the init.ora file.

6. If you are upgrading a cluster database, then set the `CLUSTER_DATABASE` initialization parameter to false. After the upgrade, you must set this initialization parameter back to `true`.

7. Shut down the instance:

```
SQL> SHUTDOWN IMMEDIATE
```

8. 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
```

9. If your operating system is UNIX, then make sure that your `ORACLE_SID` is set correctly and that the following variables point to the new release directories:
`ORACLE_HOME,PATH,ORA_NLS10,LD_LIBRARY_PATH`

10. Log in to the system as the owner of the Oracle home directory of the new Oracle Database 10g release.

11. At a system prompt, change to the `ORACLE_HOME/rdbms/admin` directory.

12. Start Up the New Database

```
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

13. If you are upgrading from a release other than 10.1, create the `SYSAUX` Tablespace. The Pre-Upgrade Information Tool provides an estimate of the minimum required size for the `SYSAUX` tablespace in the `SYSAUX` Tablespace section.

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

14. If you upgrading to release 1, run the Upgrade Script. Run the Upgrade Script corresponding to the Oracle version you would like to upgrade:
    - 8.0.6: `u0800060.sql`
    - 8.1.7: `u0801070.sql`
    - 9.0.1: `u0900010.sql`
    - 9.2: `u0902000.sql`

15. If you upgrading to Oracle Database 10g Release 2, only one common SQL script has to be invoked when performing a database upgrade. Oracle automatically determines what version is being upgraded and runs the appropriate upgrade scripts for that database and all of its included components:

```
SQL> SPOOL upgrade.log
SQL> @catupgrd.sql
```

16. Depending of the release you are upgrading to, run utlu10*s.sql (Post-Upgrade Status Tool) to display the results of the upgrade:

```
SQL> @utlu101s.sql TEXT
SQL> @utlu102s.sql
SQL> SPOOL OFF
```

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.

17. Check the spool file and verify that the packages and procedures compiled successfully. Rerun the `catupgrd.sql` script, if necessary.

18. Restart the instance

```
SQL> SHUTDOWN IMMEDIATE
SQL> STARTUP
```

19. If Oracle Label Security is in your database:

```
SQL> @olstrig.sql
```

20. Run `utlrp.sql` to recompile any remaining invalid stored PL/SQL and Java code.

```
SQL> @utlrp.sql
```

21. Verify that all expected packages and classes are valid:

```
SQL> SELECT count(*) FROM dba_objects WHERE
status='INVALID';
SQL> SELECT distinct object_name FROM
dba_objects WHERE status='INVALID';
```

22. Exit SQL*Plus

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.

Data Pump automatically selects the most appropriate access method for each table. It always tries to first use the direct-path method. Under some conditions, such as the following, it may not 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
```

Note: In Oracle 10g Release 2, a directory object named `DATA_PUMP_DIR` as created by default in the database. In Windows, it is mapped to `<ORACLE_BASE>\admin\<sid>\dpdump\` directory. By default, it is available only to privileged users.

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
- 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 Interfaces

Using the Command Line
```
expdp system/manager directory=dpump_dir1
dumpfile=expdat1.dmp
```

Using a Parameter File
```
expdp parfile=myfile.txt
```

Using Interactive-command Data Pump Export
In Data Pump export, you use the interactive-command interface 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.
This mode is enabled by pressing [Ctrl] + [C] during an export operation started with the command-line interface or the parameter file interface.

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

Data Pump Export Modes

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

Data Pump Export Parameters

File- and Directory-Related Parameters

`DIRECTORY`
specifies the location of the dump and other files.

`DUMPFILE`
provides the name of the dump file to which the export dump should be written.
You can provide multiple dump filenames in several ways:
- 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.
- using a comma-separated list.
- specifying the `DUMPFILE` 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.

Note: The directory object is not used by this parameter. The directory path is an operating system-specific directory specification. The default is the user's current directory.

`LOGFILE and NOLOGFILE`
You can use the `LOGFLE` 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:
- `ALL` exports data and definitions (metadata).
- `DATA_ONLY` exports only table rows.
- `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 used 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 Ctl+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.

```
expdp system/manager full=y
 parallel=4
 dumpfile=
   DIR1:full1%U.dat,
   DIR2:full2%U.dat,
   DIR3:full3%U.dat,
   DIR4:full4%U.dat
 filesize = 2G

impdp system/manager
 directory = MYDIR
 parallel = 4
 dumpfile = full1%U.dat,full2%U.dat,
 full3%U.dat,full4%U.dat
```

Dumpfile Compression Parameter

COMPRESSION =(METADATA_ONLY | NONE)

This parameter applies from Oracle 10.2. It specifies whether to compress metadata before writing to the dump file set. Compression reduces the amount of disk space consumed by dump files.

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=cf1n02 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.

OID

If the value is specified as n, the assignment of the exported OID during the creation of object tables and types is inhibited. Instead, a new OID is assigned. This can be useful for cloning schemas, but does not affect referenced objects. The default is y.

PCTSPACE

It accepts a greater-than-zero number. It represents the percentage multiplier used to alter extent allocations and the size of data files.

object_type: It can take one of the following values: CLUSTER,CONSTRAINT,INC_TYPE,INDEX,ROLLBACK_SEGMENT,TABLE,TABLESPACE,TYPE

```
impdp hr/hr TABLES=employees \
  DIRECTORY=dp_dir DUMPFILE=hr_emp.dmp \
  TRANSFORM=SEGMENT_ATTRIBUTES:n:table


impdp hr/hr TABLES=employees \
  DIRECTORY=dp_dir DUMPFILE=hr_emp.dmp \
  TRANSFORM=STORAGE:n:table
```

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. In this case, the external table is composed of proprietary format flat files that are operating system independent.

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

Note: You cannot use an external table population operation with an external table defined to be used with the ORACLE_LOADER access driver.

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:
- REFERENCED - Referenced columns
- 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
   DUMPFILE=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.

   Note: If you have CLOB data on a small-endian system in an Oracle database version before 10g and with a varying-width character set and you are transporting to a database in a big-endian system, the CLOB data must be converted in the destination database. RMAN does not handle the conversion during the CONVERT phase. However, Oracle database automatically handles the conversion while accessing the CLOB data.

   If you want to eliminate this run-time conversion cost from this automatic conversion, you can issue the CREATE TABLE AS SELECT command before accessing the data.

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.

Archiving Historical Data Using Transportable Tablespaces

Using Transportable Tablespaces to Perform TSPITR

Note: For information about transporting the entire database across the platforms, see the section "Cross-Platform Transportable Database".

Using Database Control to Transport Tablespaces

You can use the Transport Tablespaces wizard to move a subset of an Oracle database from one Oracle database to another, even across different platforms.

The Transport Tablespaces wizard automates the process of generating a transportable tablespace set, or integrating an existing transportable tablespace set. The wizard uses a job that runs in the Enterprise Manager job system.

You can access the wizard from the **Maintenance** | **Transport Tablespaces** link in the **Move Database Files** section.

Transport Tablespace from Backup

You can use the *transport tablespace from backup* feature to transport tablespaces at a point in time without marking the source tablespaces READ ONLY. This removes the need to set the tablespace set in READ ONLY mode while exporting its metadata which results in a period of unavailability.

The RMAN command `TRANSPORT TABLESPACE` is used to generate one version of a tablespace set. A tablespace set version comprises the following:

- The set of data files representing the tablespace set recovered to a particular point in time.
- The Data Pump export dump files generated while doing a transportable tablespace export of the recovered tablespace set
- The generated SQL script used to import the recovered tablespace set metadata into the target database. This script gives you two possibilities to import the tablespace set metadata into the target database: IMPDP or the `DBMS_STREAMS_TABLESPACE_ADM.ATTACH_TABLESPACES` procedure.

Note: this option is time-consuming compared to the method that requires setting the tablespace in READ ONLY mode.

Transport Tablespace from Backup Implementation

Following are the steps done by RMAN to implement the transport tablespace from backup:

1. While executing the `TRANSPORT TABLESPACE` command, RMAN starts an auxiliary database instance on the same machine as the source database. The auxiliary instance is started with a `SHARED_POOL_SIZE` set to 110 MB to accommodate the Data Pump needs.
2. RMAN then restores the auxiliary set as well as the recovery set by using existing backups. The restore operation is done to a point before the intended point in time of the tablespace set version.
3. RMAN recovers the auxiliary database to the specified point in time.
4. At that point, the auxiliary database is open with the `RESETLOGS` option, and `EXPDP` is used in `TRANSPORTABLE TABLESPACE` mode to generate the dump file set containing the recovered tablespace set metadata.
5. RMAN then generates the import script file that can be used to plug the tablespace set into your target database.

Note: The tablespace set may be kept online and in READ WRITE mode at the source database during the cloning process.

```
RUN {
 TRANSPORT TABLESPACE 'USERS'
   AUXILIARY DESTINATION 'C:\oraaux'
   DUMP FILE 'tbsUSERS.dmp'
   EXPORT LOG 'tbsUSERS.log'
   IMPORT SCRIPT 'imptbsUSERS.sql'
   TABLESPACE DESTINATION 'C:\oraaux\ttbs'
   UNTIL TIME "to_date('28-04-2007
14:05:00','dd-mm-yyyy, HH24:MI:SS')";}
```

`DUMP FILE`
specifies the name of the generated Data Pump export dump file. Its default value is `dmpfile.dmp`

`EXPORT LOG`
specifies the name of the log file for the Data Pump export job. Its default value is `explog.log`

`IMPORT SCRIPT`
specifies the name of the sample import script. Its default value is `impscrpt.sql`. The import script is written to the location specified by the `TABLESPACE DESTINATION` parameter.

`TABLESPACE DESTINATION`
it is a required parameter that specifies the default location for the data files in the recovery set.

`UNTIL`
The `UNTIL` clause is used to specify the point-in-time for the tablespace set version. You may specify the point-in-time as an SCN, TIME, or log SEQUENCE.

Versioning Tablespaces

In Oracle Database 10g Release 2, you can build a repository to store versions of tablespace, referred to as a *tablespace rack*. The repository may be located in the same database as the tablespaces being versioned, or may be located in a different database. Handling this option is not covered in this document.

Loading Data from Flat Files by Using EM

The new Load Data wizard enhancements enable you to load data from external flat files into the database. It uses the table name and data file name that you specify, along with other information, to scan the data file and define a usable SQL*Loader control file. The wizard will create the control file for you. It then uses SQL*Loader to load the data.

Note: Not all control file functionality is supported in the Load Data wizard.

You can access the Load Data page from: **Maintenance tabbed page** | **Move Row Data** section

DML Error Logging Table

DML Error Logging Table

This feature (in Release 2) allows bulk DML operations to continue processing, when a DML error occurs, with the ability to log the errors in a DML error logging table.

DML error logging works with `INSERT`, `UPDATE`, `MERGE`, and `DELETE` statements.

To insert data with DML error logging:

1. Create an error logging table.

   This can be automatically done by the `DBMS_ERRLOG.CREATE_ERROR_LOG` procedure. It creates an error logging table with all of the mandatory error description columns plus all of the columns from the named DML table.

   ```
   DBMS_ERRLOG.CREATE_ERROR_LOG(<DML
   table_name>[,<error_table_name>])
   ```

   default logging table name is ERR$_ plus first 25 characters of table name

   You can create the error logging table manually using the normal DDL statements but it must contain the following mandatory columns:

   ```
   ORA_ERR_NUMBER$     NUMBER
   ORA_ERR_MESG$       VARCHAR2(2000)
   ORA_ERR_ROWID$      ROWID
   ORA_ERR_OPTYP$      VARCHAR2(2)
   ORA_ERR_TAG$        VARCHAR2(2000)
   ```

2. Execute an `INSERT` statement and include an error logging clause.

   ```
   LOG ERRORS [INTO <error_table>] [('<tag>')]
   [REJECT LIMIT <limit>]
   ```

   If you do not provide an error logging table name, the database logs to an error logging table with a default name.

   You can also specify `UNLIMITED` for the `REJECT LIMIT` clause. The default reject limit is zero, which means that upon encountering the first error, the error is logged and the statement rolls back.

```
DBMS_ERRLOG.CREATE_ERROR_LOG('DW_EMPL')
INSERT INTO dw_empl
  SELECT employee_id, first_name, last_name,
hire_date, salary, department_id
  FROM employees
  WHERE hire_date > sysdate - 7
  LOG ERRORS ('daily_load') REJECT LIMIT 25
```

Asynchronous Commit

In Oracle 10.2 `COMMITs` can be optionally deferred.

This eliminates the wait for an I/O to the redo log but the system must be able to tolerate loss of asynchronously committed transaction.

```
COMMIT [ WRITE [ IMMEDIATE|BATCH] [WAIT |
NOWAIT]
```

`IMMEDIATE` specifies redo should be written immediately by LGWR process when transaction is committed (default)

`BATCH` causes redo to be buffered to redo log

`WAIT` specifies commit will not return until redo is persistent in online redo log (default)

`NOWAIT` allows commit to return before redo is persistent in redo log

```
COMMIT;                 --  =IMMEDIATE WAIT
COMMIT WRITE;           --  = COMMIT;
COMMIT WRITE IMMEDIATE;--  = COMMIT;
COMMIT WRITE IMMEDIATE WAIT;  -- = COMMIT;
COMMIT WRITE BATCH;           -- = BATCH WAIT
COMMIT WRITE BATCH NOWAIT;    -- = BATCH NOWAIT
```

`COMMIT_WRITE` initialization parameter determines default value of `COMMIT WRITE` statement.

Can be modified using `ALTER SESSION` statement

```
ALTER SESSION SET COMMIT_WRITE = 'BATCH,NOWAIT'
```

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 intialization 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 (ASMM)

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:

- Buffer cache (DB_CACHE_SIZE)
- Shared pool (SHARED_POOL_SIZE)
- Large pool (LARGE_POOL_SIZE)
- Java pool (JAVA_POOL_SIZE)
- Streams pool (STREAMS_POOL_SIZE) in *Release 2*

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:

- Log buffer
- Other buffer caches, such as KEEP, RECYCLE, and other block sizes
- Streams pool (in Release 1 only)
- Fixed SGA and other internal allocations
- 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$SGA_DYNAMIC_COMPONENTS view to see the values assigned to the auto-tuned components. Whereas the V$PARAMETER will display the value you *set* to the auto-tuned SGA parameter, not the value assigned by the ASMM.

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.

```
COLUMN COMPONENT FORMAT A30
SELECT COMPONENT , CURRENT_SIZE/1024/1024 MB
FROM V$SGA_DYNAMIC_COMPONENTS
WHERE CURRENT_SIZE <>0
```

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.

Changes on DBMS_STATS

Oracle Database 10g introduces new values for the GRANULARITY and DEGREE arguments of the GATHER_*_STATS procedures to simplify the determination of the calculated statistics. Unless you are an experienced user, you should use the new default values:

- GRANULARITY
  - AUTO (default): The procedure determines the granularity based on the partitioning type. It collects the global-, partition-, and subpartition-level statistics if the subpartitioning method is LIST. Otherwise, it collects only the global- and partition-level statistics.
  - GLOBAL AND PARTITION: Gathers the global- and partition-level statistics. No subpartition-level statistics are gathered even if it is a composite partitioned object.
- DEGREE
  - AUTO_DEGREE**:** This value enables the Oracle server to decide the degree of parallelism automatically. It is either 1 (serial execution) or DEFAULT_DEGREE (the system default value based on the number of CPUs and initialization parameters) according to the size of the object.

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, if the STATISTICS_LEVEL initialization parameter is set to TYPICAL or ALL.

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

- LOCK_TABLE_STATISTICS
- UNLOCK_TABLE_STATISTICS
- LOCK_SCHEMA_STATISTICS
- 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 your 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.

Rule-Based Optimizer Obsolescence

RBO still exists in Oracle Database 10g but is an unsupported feature. No code changes have been made to RBO, and no bug fixes are provided.

Database and Instance Level Trace

In Oracle 10.2 includes new procedures to enable and disable trace at database and/or instance level for a given Client Identifier, Service Name, MODULE and ACTION.

To enable trace in the whole database

```
DBMS_MONITOR.DATABASE_TRACE_ENABLE
```

To enable trace in the instance level

```
DBMS_MONITOR.DATABASE_TRACE_ENABLE
(INSTANCE_NAME=>'RAC1')
```

This procedure disables SQL trace for the whole database or a specific instance

```
DBMS_MONITOR.DATABASE_TRACE_DISABLE(
instance_name  IN VARCHAR2 DEFAULT NULL)
```

For information about tracing at service level, refer to the section "Enhancements in Managing Multitier Environments".

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:

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

The Undo Advisor

This OEM utility provides you undo related functions like:

- undo tablespace size recommendations
- 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
```

Automatically Tuned Multiblock Reads

The DB_FILE_MULTIBLOCK_READ_COUNT parameter controls the number of blocks prefetched into the buffer cache during scan operations, such as full table scan and index fast full scan.

Oracle Database 10g Release 2 automatically selects the appropriate value for this parameter depending on the operating system optimal I/O size and the size of the buffer cache.

This is the default behavior in Oracle Database 10g Release 2, if you do not set any value for DB_FILE_MULTIBLOCK_READ_COUNT parameter, or you explicitly set it to 0. If you explicitly set a value, then that value is used, and is consistent with the previous behavior.

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:

- computing metrics
- 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_SETTIN
GS( 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

`DBA_HIST_SNAPSHOT`	shows all snapshots saved in the AWR.
`DBA_HIST_WR_CONTROL`	displays the settings to control the AWR.
`DBA_HIST_BASELINE`	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.

Generate ASH Reports

In Oracle Release 2, you can generate ASH Report. This is a digest of the ASH samples that were taken during a time period. Some of the information it shows are top wait events, top SQL, top SQL command types, and top sessions, among others.

On Database Control:

Performance -> Run ASH Report button

On SQL* Plus:

Run the following script
`$ORACLE_HOME/rdbms/admin/ashrpt.sql`

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`):

- Event Class Metrics
- Event Metrics
- File Metrics
- Service Metrics
  `V$SERVICEMETRIC, V$SERVICEMETRIC_HISTORY`
- Session Metrics
- System Metrics
  `V$SYSMETRIC, V$SYSMETRIC_HISTORY`
- 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:

- `CPU_TIME_PER_CALL` specifies the metric identifier. For a list of support metrics, see PL/SQL Packages and Types Reference.
- 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.
- 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.
- The name of the instance is set to inst1.
- 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)
```

See the section "Proactive Tablespace Management" for more examples of using `DBMS_SERVER_ALERT` package.

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_QUE` 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_QUE`.
4. To read an alert message, you can use the `DBMS_AQ.DEQUEUE` procedure or `OCIAQDeq` 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

DBA_THRESHOLDS	lists the threshold settings defined for the instance.
DBA_OUTSTANDING_ALERTS	describes the outstanding alerts in the database.
DBA_ALERT_HISTORY	lists a history of alerts that have been cleared.
V$ALERT_TYPES	provides information such as group and type for each alert.
V$METRICNAME	contains the names, identifiers, and other information about the system metrics.
V$METRIC and V$METRIC_HISTORY	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.

Adaptive Thresholds

New in Oracle Database 10g Release 2, adaptive thresholds use statistical measures of central tendency and variability to characterize normal system behavior and trigger alerts when observed behavior deviates significantly from the norm.

As a DBA, you designate a period of system time as a metric baseline which should represent the period of normal activity of your system. This baseline is then divided into time groups. You can specify critical and warning thresholds relative to the computed norm.

Metric Baselines and Thresholds Concepts

Metric baselines are of two types:

- **Static baselines** are made up of a single user-defined interval of time.
- **Moving window** baselines are based on a simple functional relationship relative to a reference time. They are currently defined as a specific number of days from the past.

Two types of adaptive thresholds are supported:

- **Significance level thresholds**: The system can dynamically set alert thresholds to values representing statistical significance as measured by the active baseline. Alerts generated by observed metric values exceeding these thresholds are assumed to be unusual events and, therefore, possibly indicative of, or associated with, problems.
- **Percent of maximum thresholds**: You can use this type of threshold to set metric thresholds relative to the trimmed maximum value measured over the baseline period and time group. This is most useful when a static baseline has captured some period of specific workload processing and you want to signal when values close to or exceeding peaks observed over the baseline period.

Metric Baselines and Time Groups

The supported time grouping schemes have the daily and weekly options.

The **daily options** are:

- **By hour of day**: Aggregate each hour separately for strong variations across hours.
- **By day and night**: Aggregate the hours of 7:00 a.m. to 7:00 p.m. as day and 7:00 p.m. to 7:00 a.m. as night.
- **By all hours**: Aggregate all hours together when there is no strong daily cycle.

The **weekly time** grouping options are:

- **By day of week**: Aggregate days separately for strong variations across days.
- **By weekday and weekend**: Aggregate Monday to Friday together and Saturday and Sunday together.
- **By all days:** Aggregate all days together when there is no strong weekly cycle.

Enabling Metric Baselining

Before you can successfully use metric baselines and adaptive thresholds, you must enable that option by using Enterprise Manager. Internally, Enterprise Manager causes the system metrics to be flushed, and submits a job once a day that is used to compute moving-window baseline statistics. It also submits one job once every hour to set thresholds after a baseline is activated.

You can enable metric baselining from the **Database Home page** | **Related Links** | **Metric Baselines** | **Enable Metric Baselines**

Activating the Moving Window Metric Baseline

Use the **Metric Baselines** page to configure your active baseline.

After baselining is activated, you can access the Metric Baselines page directly from the Database Home page by clicking the Metric Baselines link in the Related Links section.

You can either use one *Moving window* metric baseline or select an already defined *Static baseline*.

When using a Moving Window baseline, you need to select the time period you want to define for this baseline, such as “Trailing 7 days.” This period moves with the current time. The most recent seven-day period becomes the baseline period (or reference time) for all metric observations and comparisons today. Tomorrow, this reference period drops the oldest day and picks up today.

Then, define the *Time Grouping* scheme. Grouping options available for a baseline depend on the size of the time period for the baseline. The system automatically gives you realistic choices.

After this is done, click Apply. Enterprise Manager computes statistics on all the metrics referenced by the

baseline. The computing of statistics is done everyday automatically.

Setting Adaptive Alert Thresholds

Use the Edit Baseline Alert Parameters page to:

- View the current status of the 15 metrics that can be set with adaptive thresholds
- Set thresholds for Warning Level, Critical Level, and Occurrences
- Specify threshold action for insufficient statistical data

You can visualize the collected statistics for your metric baselines by following the links: **Metric Baselines | click Set Adaptive Thresholds after selecting the corresponding baseline | Manage Adaptive Thresholds | click the corresponding eyeglasses icon in the Details column**

Creating Static Metric Baselines

Follow the links: **Manage Static Metric Baselines** link in the Related Links section | **Create Static Metric Baseline**

On the Create Static Metric Baseline page, specify a Name for your static metric baseline. Then select a Time Period by using the Begin Day and End Day fields. These two dates define the fixed interval that calculates metric statistics for later comparisons. After this is done, select the Time Grouping scheme:

- **By Hour of Day**: Creates 24 hourly groups
- **By Day and Night**: Creates two groups: day hours (7:00 a.m. to 7:00 p.m.) and night hours (7:00 p.m. to 7:00 a.m.).
- **By Day of Week**: Creates seven daily groups.
- **By Weekdays and Weekend**: Creates two groups: weekdays (Monday through Friday) together and weekends (Saturday and Sunday) together.

You can combine these options. For instance, grouping by Day and Night and Weekdays and Weekend produces four groups.

Then, click Compute Statistics to compute statistics on all the metrics referenced by the baseline. Enterprise Manager computes statistics only once, which is when the baseline is created.

If an alert message appears in the Model Fit column, either there is insufficient data to perform reliable calculations, or the data characteristics do not fit the metric baselines model.

If there is insufficient data to reliably use statistical alert thresholds, either extend the time period or make time groups larger to aggregate statistics across larger data samples.

Considerations

- Baselining must be enabled using Enterprise Manager.
- Only one moving window baseline can be defined.
- Multiple static baselines can be defined.
- Only one baseline can be active at a time.
- Adaptive thresholds require an active baseline.

Metric value time series can be normalized against a baseline by converting each observation to some integer measure of its statistical significance relative to the baseline.

You can see the normalized view of your metrics on the Baseline Normalized Metrics page. You access this page from the Metric Baselines page by clicking the **Baseline Normalize Metrics** link in the Related Links section.

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 `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=>'sele
ct 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.

Performance Pages in 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 a dynamic initialization parameter and its default value is TRUE. This setting disables error reporting of indexes and index partitions marked as UNUSABLE.

Note: This setting does not disable error reporting for unusable indexes that are unique because allowing insert and update operations on the table might violate the corresponding constraint.

Note: The database still records an alert message in the alert.log file whenever an index is marked as unusable.

Using Hash-Partitioned Global Indexes

- In Oracle 10g, you can create hash-partitioned global indexes. (Previous releases support only range-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 PARTITITON`, 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)
```

- To `add a new index partition`

```
ALTER INDEX sales_hash ADD PARTITION p4
TABLESPACE tbs_4 [PARALLEL]
```

Notice the following for the previous command:

  - The newly added partition is populated with index entries rehashed from an existing partition of the index as determined by the hash mapping function.
  - If a partition name is not specified, a system-generated name of form `SYS_P###` is assigned to the index partition.
  - If a tablespace name is not specified, the partition is placed in a tablespace specified in the index-level `STORE IN` list, or user, or system default tablespace, in that order.

- To reverse adding a partition, or in other words to reduce by one the number of index partitions, you coalesce one of the index partitions then you destroy it. Coalescing a partition distributes index entries of an index partition into one of the index partitions determined by the hash function.

```
ALTER INDEX sales_hash COALESCE PARTITION
PARALLEL
```

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. This was not available in previous versions.
- 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))
```

Bitmap Index Storage Enhancements

Oracle Database 10g provides enhancements for handling DML operations involving bitmap indexes. These improvements eliminate the slowdown of bitmap index performance, which occurs under certain DML situations. Bitmap indexes now perform better and are less likely to be fragmented when subjected to large volumes of single-row DML operations.

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. This also applies in the undo tablespace.
- 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.

Examples:

To set your own databasewide default threshold values for the Tablespace Space Usage metric:

```
EXECUTE DBMS_SERVER_ALERT.SET_THRESHOLD(
 METRICS_ID=>dbms_server_alert.tablespace_pct_f
 ull,
 WARNING_OPERATOR=>dbms_server_alert.operator_g
 e,
 WARNING_VALUE=>80,
 CRITICAL_OPERATOR=>dbms_server_alert.operator_
 ge,
 CRITICAL_VALUE=>95,
 OBSERVATION_PERIOD=>1,
 CONSECUTIVE_OCCURRENCES=>1,
 INSTANCE_NAME=>NULL,
 OBJECT_TYPE=>dbms_server_alert.object_type_tab
 lespace,
 OBJECT_NAME=>NULL)
```

To set a warning threshold of 80% and a critical threshold of 95% on the EXAMPLE tablespace, use the same previous example except `OBJECT_NAME` parameter should take value of 'EXAMPLE'

To turn off the space-usage tracking mechanism for the EXAMPLE tablespace:

```
EXECUTE DBMS_SERVER_ALERT.SET_THRESHOLD(
```

```
METRICS_ID=>dbms_server_alert.tablespace_pct_f
ull,
WARNING_OPERATOR=>dbms_server_alert.operator_d
o_not_check,
WARNING_VALUE=>'0',
CRITICAL_OPERATOR=>dbms_server_alert.operator_
do_not_check,
CRITICAL_VALUE=>'0',
OBSERVATION_PERIOD=>1,
CONSECUTIVE_OCCURRENCES=>1,
INSTANCE_NAME=>NULL,
OBJECT_TYPE=>dbms_server_alert.object_type_tab
lespace,
OBJECT_NAME=>'EXAMPLE')
```

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.
- In Oracle 10.2 you can also shrink:
  - LOB Segments
  - Function Based Indexes
  - IOT Overflow Segments

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
```

To shrinks a LOB segment:

```
ALTER TABLE employees MODIFY LOB(resume)
(SHRINK SPACE)
```

To shrink an IOT overflow segment belonging to the EMPLOYEES table:

```
ALTER TABLE employees OVERFLOW SHRINK SPACE
```

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.

Automatic Segment Advisor

Automatic Segment Advisor is implemented by the `AUTO_SPACE_ADVISOR_JOB` job. This job executes the `DBMS_SPACE.AUTO_SPACE_ADVISOR_JOB_PROC` procedure at predefined points in time.

When a Segment Advisor job completes, the job output contains the space problems found and the advisor recommendations for resolving those problems.

You can view all Segment Advisor results by navigating to the **Segment Advisor Recommendations** page. You access this page from the home page by clicking the **Segment Advisor Recommendations** link in the **Space Summary** section.

The following views display information specific to Automatic Segment Advisor:

- `DBA_AUTO_SEGADV_SUMMARY`: Each row of this view summarizes one Automatic Segment Advisor run. Fields include number of tablespaces and segments processed, and number of recommendations made.
- `DBA_AUTO_SEGADV_CTL`: This view contains control information that Automatic Segment Advisor uses to select and process segments.

Object Size Growth Analysis

You plan to create a table in a tablespace and populate it with data. So, you want to estimate its initial size. This can be achieved using Segment Advisor in the EM or its package `DBMS_SPACE`.

Estimating Object Size using EM

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

Follow these steps:

1. From the Database Control home page, click the **Administration** tab.
2. Under the **Storage** section, click the **Tables** link.
3. Click the **Create** button to create a new table.
4. 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.
5. 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.

Estimating Object Size using DBMS_SPACE

For example, if your table has 30,000 rows, its average row size is 3 and the PCTFREE parameter is 20. You can issue the following code:

```
set serveroutput on
DECLARE
 V_USED NUMBER;
 V_ALLOC NUMBER;
BEGIN
 DBMS_SPACE.CREATE_TABLE_COST (
  TABLESPACE_NAME => 'USERS',
  AVG_ROW_SIZE => 30,
  ROW_COUNT => 30000,
  PCT_FREE => 5,
  USED_BYTES => V_USED,
  ALLOC_BYTES => V_ALLOC);
 DBMS_OUTPUT.PUT_LINE('USED: '|| V_USED/1024 ||
' KB');
 DBMS_OUTPUT.PUT_LINE('ALLOCATED: '||
V_ALLOC/1024 || ' KB');
END;
```

The USED_BYTES represent the actual bytes used by the data. The ALLOC_BYTES represent the size of the table when it is created in the tablespace. This takes into account, the size of the extents in the tablespace and tablespace extent management properties.

If you want to make the estimation based on the column definitions (not average row size and PCTFREE):

```
set serveroutput on
DECLARE
 UB NUMBER;
 AB NUMBER;
 CL SYS.CREATE_TABLE_COST_COLUMNS;
BEGIN
 CL := SYS.CREATE_TABLE_COST_COLUMNS(
  SYS.CREATE_TABLE_COST_COLINFO('NUMBER',10),
  SYS.CREATE_TABLE_COST_COLINFO('VARCHAR2',30),
  SYS.CREATE_TABLE_COST_COLINFO('VARCHAR2',30),
  SYS.CREATE_TABLE_COST_COLINFO('DATE',NULL));
  DBMS_SPACE.CREATE_TABLE_COST('USERS',CL,100000
  ,0,UB,AB);
  DBMS_OUTPUT.PUT_LINE('USED: '|| UB/1024 || '
  KB');
   DBMS_OUTPUT.PUT_LINE('ALLOCATED: '|| AB/1024
  || ' KB');
END;
```

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:

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

Rollback Monitoring

In Oracle Database 10g, when a transaction rolls back, the event is recorded in the view V$SESSION_LONGOPS, if the process takes more than six seconds. This view enables you to estimate when the monitored rollback process will finish.

```
SELECT TIME_REMAINING,
       SOFAR/TOTALWORK*100 PCT
FROM   V$SESSION_LONGOPS
WHERE  SID = 9
  AND  OPNAME ='Transaction Rollback'
```

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 occcupants 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 deftbs 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

1. Database Control home page, Administration, Storage Section, Tablespaces.
2. 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.

It has the following advantages:

- You define more than one default temporary tablespace, and a single SQL operation can use more than one temporary tablespace for sorting. This prevents large tablespace operations from running out of temporary space.
- Enables one particular user to use multiple temporary tablespaces in different sessions at the same time
- Enables the slave processes in a single parallel operation to use multiple 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:

- DBA_TABLESPACE_GROUPS
- DBA_USERS

Bigfile Tablespaces

- A bigfile tablespace (BFT) contains only one very large file (can be as large as 8 to 128 terabytes depending on block size).
- The main benefit is easier management of tablespaces and their datafiles in very large databases (VLDB). All operations that were performed on data files in previous releases can now be performed on BFT tablespaces. For example: ALTER TABLESPACE … RESIZE

Big File Teblespaces 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.
- You cannot change tablespace type from smallfile to bigfile or vice versa. However, you can migrate object between tablespace types by using either the ALTER TABLE ... MOVE or CREATE TABLE ... AS
- To avoid performance implications, use the following table as a guide to the maximum number of extents for a BFT with specific block size. If the expected size requires more extents than specified in the table, you can create the tablespace with UNIFORM option (instead of AUTOALLOCATE) with a large extend size.

Database Block Size	Recommended Maximum Number of Extents
2 KB	100,000
4 KB	200,000
8 KB	400,000
16 KB	800,000

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;
```

You can view the default tablespace type using the following command:

```
SELECT PROPERTY_VALUE
 FROM DATABASE_PROPERTIES
 WHERE PROPERTY_NAME='DEFAULT_TBS_TYPE'
```

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`:

- `DBA_TABLESPACES`
- `USER_TABLESPACES`
- `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
```

Note: The functions `DATA_BLOCK_ADDRESS_FILE` and `DATA_BLOCK_ADDRESS_BLOCK` in the package `DBMS_UTILITY` do not return the expected results with BFTs.

Bigfile Tablespaces and DBVERIFY

You cannot run multiple instances of DBVERIFY utility in parallel against BFT. However, integrity-checking parallelism can be achieved with BFTs by starting multiple instances of DBVERIFY on parts of the single large file. In this case, you have to explicitly specify the starting and ending block addresses for each instance.

```
dbv FILE=BFile1 START=1 END=10000
dbv FILE=BFile1 START=10001
```

Viewing Tablespace Contents

You can obtain detailed information about the segments in each tablespace using Enterprise Manager.

On the **Tablespaces** page, select the tablespace of interest, choose Show **Tablespace Contents** from the Actions drop-down list, and click **Go**. The Processing: Show Tablespace Contents page is displayed.

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;
```

`SINGLE TABLE`	indicates that the cluster is a type of hash cluster containing only one table.
`HASH IS expr`	Specifies an expression to be used as the hash function for the hash cluster.
`HASHKEYS`	this clause creates a hash cluster and specify the number of hash values for the hash cluster.
`SIZE`	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)
```

Partitioned IOT Enhancements

The following are the newly supported options for partitioned index-organized tables (IOTs):

- **List-partitioned IOTs**: All operations allowed on list-partitioned tables are now supported for IOTs.
- **Global index maintenance**: With previous releases of the Oracle database, the global indexes on

partitioned IOTs were not maintained when partition maintenance operations were performed. After DROP, TRUNCATE, or EXCHANGE PARTITION, the global indexes became UNUSABLE. Other partition maintenance operations such as MOVE, SPLIT, or MERGE PARTITION did not make the global indexes UNUSABLE, but the performance of global index–based access was degraded because the guess–database block addresses stored in the index rows were invalidated. Global index maintenance prevents these issues from happening, keeps the index usable, and also maintains the guess–data block addresses.

- **Local partitioned bitmap indexes**: The concept of a mapping table is extended to support a mapping table that is equi-partitioned with respect to the base table. This enables the creation of bitmap indexes on partitioned IOTs.
- **LOB columns** are now supported in all types of partitioned IOTs.

Redefine a Partition Online

The package DBMS_REDEFINITION is known to be used as a tool to change the definition of the objects while keeping them accessible (online). In previous versions, if you use it to move a partitioned table to another tablespace, it will move the entire table. This results in massive amount of undo and redo generation.

In Oracle 10g, you can use the package to move a single partition (instead of the entire table). The following code illustrates the steps you follow.

1. Confirm that you can redefine the table online. Having no output after running the following code means the online redefinition is possible:

```
BEGIN
 DBMS_REDEFINITION.CAN_REDEF_TABLE(
 UNAME => 'HR',
 TNAME => 'customers',
 OPTIONS_FLAG =>
DBMS_REDEFINITION.CONS_USE_ROWID,
 PART_NAME    => 'p1');
END;
```

2. Create a temporary (interim) table to hold the data for that partition:

```
CREATE TABLE hr.customers_int
 TABLESPACE custdata
 AS
 SELECT * FROM hr.customers
 WHERE 1=2;
```

Note: If the table customers had some local indexes, you should create those indexes (as non-partitioned, of course) on the table customers_int.

3. Start the redefinition process:

```
BEGIN
 DBMS_REDEFINITION.START_REDEF_TABLE(
  UNAME => 'HR',
  ORIG_TABLE => 'customers',
  INT_TABLE  => 'customers_int',
  PART_NAME  => 'p1' ); -- partition to move
END;
```

4. If there were DML operations against the table during the move process, you should synchronize the interim table with the original table:

```
BEGIN
 DBMS_REDEFINITION.SYNC_INTERIM_TABLE (
 UNAME => 'HR',
 ORIG_TABLE => 'customers',
 INT_TABLE  => 'customers_int',
 COL_MAPPING  => NULL,
 OPTIONS_FLAG =>
DBMS_REDEFINITION.CONS_USE_ROWID,
 PART_NAME  => 'p1' );
END;
```

5. Finish the redefinition process:

```
BEGIN
 DBMS_REDEFINITION.FINISH_REDEF_TABLE (
 UNAME => 'HR',
 ORIG_TABLE => 'customers',
 INT_TABLE  => 'customers_int',
 PART_NAME  => 'p1');
END;
```

To confirm the partition P1 was moved to the new tablespace:

```
 SELECT PARTITION_NAME, TABLESPACE_NAME,
 NUM_ROWS
 FROM USER_TAB_PARTITIONS
 WHERE PARTITION_NAME='P1'
```

Note: If there is any global index on the table, they will be marked as UNUSABLE and must be rebuilt.

Note: You cannot change the structure of the table during the definition process.

Note: statistics of object moved with this tool are automatically generated by end of the process.

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.

If you are copying a database datafile, do not forget to make it READ ONLY before you start to copy.

You can monitor copying progress using V$SESSION_LONGOPS view.

Dropping Partitioned Table

In previous versions, if you drop a partitioned table, Oracle removes all the partitions at once. This led to a time and resource consuming process.

In Oracle Database 10g Release 2, when you drop a partitioned table, partitions are dropped one by one. Because each partition is dropped individually, fewer resources are required than when the table is dropped as a whole.

Dropping Empty Datafiles

In Oracle 10g release 2, empty datafiles can be dropped

```
ALTER TABLESPACE test DROP DATAFILE 'hr1.dbf';
```

You cannot drop non-empty datafiles

```
ORA-03262: the file is non-empty
```

You cannot drop first file in tablespace

```
ORA-03263: cannot drop the first file of
tablespace HR
```

Renaming Temporary Files

In Oracle 10.2 temporary files can be renamed.

```
ALTER DATABASE TEMPFILE 'temp1.dbf' OFFLINE

$ mv temp1.dbf temp2.dbf

ALTER DATABASE RENAME FILE 'temp1.dbf' TO
'temp2.dbf'

ALTER DATABASE TEMPFILE 'temp1.dbf' ONLINE
```

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.
- You can create time-based or event-based jobs. Events can be application-generated or scheduler-generated.
- 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; INTERVAL=10	executes a job every 10 days
FREQ=HOURLY; INTERVAL=2	executes a job every other hour
FREQ=WEEKLY; BYDAY=FRI	executes a job every Friday.
FREQ=WEEKLY; INTERVAL=2; BYDAY=FRI	executes a job every other Friday.
FREQ=MONTHLY; BYMONTHDAY=1	executes a job on the last day of the month
FREQ=YEARLY; BYMONTH=DEC; BYMONTHDAY=31	executes a job on the 31st of December.
FREQ=MONTHLY; BYDAY=2FRI	executes a job every second Friday of the month

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 calendering 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')
```

START_DATE	Time when the Window will open.
REPEAT_INTERVAL	The next time the window will open again.
RESOURCE_PLAN	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 TEST_RESOURCEPLAN.
DURATION	Window will remain open for a period of 60 minutes, after which it will close.
END_DATE	Window will open for the last time on December 31, 2005, after which it will be disabled and closed.
WINDOW_PRIORITY	Possible values are: LOW, HIGH. 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 calendering 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.

Event-Based Scheduling

- Jobs can be triggered based on events. An application can notify the Scheduler to start a job by enqueuing a message onto an Oracle Streams AQ queue. In other words, the job runs when the event is raised.
- There are two types of events:
  - **User- or application-generated** events: An application can raise an event to be consumed by the Scheduler. The Scheduler reacts to the event by starting a job. Example of such events: a running job completes; a file arrives on the file system; an account within the database is locked; and the inventory reaches a low threshold.
  - **Scheduler-generated** events: The Scheduler can raise an event to indicate state changes that occur within the Scheduler itself. For example, the Scheduler can raise an event when a job starts, when a job completes, when a job exceeds its allotted run time, and so on.

To create an event-based job, you must set these two attributes with the CREATE_JOB procedure:

- queue_spec: A queue specification that includes the name of the queue where your application enqueues messages to raise job start events, or in the case of a secure queue, the <queue_name>, <agent_name> pair
- event_condition: A conditional expression based on message properties that must evaluate to TRUE for the message to start the job. The expression must use the same syntax as an Oracle Streams AQ rule condition. You can include user data properties in the expression, provided that the message payload is a user-defined object type, and that you prefix object attributes in the expression with tab.user_data.

For more information about how to create queues and enqueue messages, refer to the Oracle Streams Advanced Queuing User's Guide and Reference documentation.

Events Raised by the Scheduler

First you must create the job by using the CREATE_JOB procedure and then use the SET_ATTRIBUTE procedure to modify the attribute's default value. The Scheduler then raises the events by enqueuing messages onto the default event queue SYS.SCHEDULER$_EVENT_QUEUE.

The queue is based on the SCHEDULER$_EVENT_INFO type, which contains the following attributes: event_type,object_owner, object_name, event_timestamp, error_code, error_msg, event_status, log_id, run_count and failure_count.

The event type can be one of the following:

- JOB_START: A job has started for the first time, or a job was started on a retry attempt. To determine which is the case, you can use the EVENT_STATUS field for further details: 0x01 - normal start, 0x02 - retry
- JOB_SUCCEEDED
- JOB_FAILED: The job resulted in an error or was not able to run due to process death or database shutdown. The EVENT_STATUS field indicates the cause of failure: 0x04: Error during job execution, 0x08: Slave crash or forced shutdown
- JOB_BROKEN: The job is marked broken after retrying unsuccessfully.
- JOB_COMPLETED: The job has a status of COMPLETED after it has reached its maximum number of runs or its end date.
- JOB_STOPPED: The job terminated normally after a soft or hard kill was issued. The EVENT_STATUS field indicates how the job was stopped: 0x10 - Stop without force, 0x20 - Stop with force
- JOB_OVER_MAX_DUR: The job has been running for a longer amount of time than was specified by the job max_run_duration attribute.
- JOB_SCH_LIM_REACHED: The schedule limit for a job has been exceeded and the job has been rescheduled.

```
DBMS_SCHEDULER.SET_ATTRIBUTE('hr.do_backup',
'raise_events', DBMS_SCHEDULER.JOB_FAILED)
```

```
DBMS_SCHEDULER.CREATE_JOB(
  job_name=>'ADMIN.REPORT_FAILED_BACKUP',
  job_type => 'STORED_PROCEDURE',
```

```
  job_action => 'ADMIN.REPORT_BACKUP_FAILURE',
  start_date => SYSTIMESTAMP,
  event_condition =>
   'tab.user_data.object_owner = ''HR'' and
tab.user_data.object_name = ''DO_BACKUP''
    and tab.user_data.event_type
='DBMS_SCHEDULER.JOB_FAILED',
  queue_spec =>
'SYS.SCHEDULER$_EVENT_QUEUE,QUEUE_AGT')
```

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.

Scheduler Job Chain

A chain is a named series of programs that are linked together for a combined objective. Each position within a chain of interdependent programs is referred to as a step. Each step can point to one of the following: a program, another chain (a nested chain), an event.

Note: This feature introduced in Oracle 10g release 2.

To create and use a chain:

1. Create a chain object

```
DBMS_SCHEDULER.CREATE_CHAIN (
  CHAIN_NAME  => 'bulk_load_chain',
  RULE_SET_NAME  => NULL,
  EVALUATION_INTERVAL => NULL,
  COMMENTS  => 'Load data and run reports')
```

2. Define one or more chain steps. You define a step that points to a program or nested chain.

```
DBMS_SCHEDULER.DEFINE_CHAIN_STEP (
 CHAIN_NAME  => 'bulk_load_chain',
 STEP_NAME  => 'do_bulk_load',
 PROGRAM_NAME => 'hr.load_data_prog)
```

Also you can define a step that waits for an event to occur by using the `DEFINE_CHAIN_EVENT_STEP` procedure. Procedure arguments can point to an event schedule or can include an in-line queue specification and event condition.

```
DBMS_SCHEDULER.DEFINE_CHAIN_EVENT_STEP (
   CHAIN_NAME  => 'bulk_load_chain',
   STEP_NAME  => 'stop_when_disk_full_evt'
   EVENT_SCHEDULE_NAME => 'disk_full_sched')
```

```
DBMS_SCHEDULER.DEFINE_CHAIN_EVENT_STEP (
   CHAIN_NAME  => 'bulk_load_chain',
   STEP_NAME  => 'load_data_evt',
   EVENT_CONDITION =>
'tab.user_data.object_owner=''HR'' and
tab.user_data.object_name = ''DATA.TXT'' and
tab.user_data.event_type =''FILE_ARRIVAL'' ',
   QUEUE_SPEC => 'HR.LOAD_JOB_EVENT_Q')
```

3. Define chain rules. Each rule has a *condition* and an *action*.

If the condition evaluates to TRUE, the action is performed. Conditions are usually based on the outcome of one or more previous steps. A condition accepts Boolean and numeric integer values in an expression. The entire expression must evaluate to a Boolean value.

The simplified syntax of a chain condition is as follows:

```
'factor|NOT(factor)[AND|OR factor]'
```

factor:

```
stepname ERROR_CODE number|[NOT]step_condition
```

When creating a rule condition using the simplified syntax:

- You specify one or more factors, and a Boolean operator (AND, OR, or NOT).
- A factor can be either a simple Boolean value (TRUE or FALSE) or a chain condition. A chain condition describes the condition of another step in the job chain. You can use the following to describe the chain condition:
  - The current state of the chain step:
    - `SUCCEEDED`
    - `FAILED`
    - `STOPPED`
    - `COMPLETED`
  - The error code returned by the chain step. The error is a numeric value, and can be:
    - Evaluated within a numeric clause
    - Compared to a list of values using an IN clause

You can use negative factors, by enclosing the factor in parentheses and prefixing the factor with the NOT operator.

Examples:

```
'step1 SUCCEEDED AND step2 ERROR_CODE = 3'
'TRUE'
'step3 NOT COMPLETED AND NOT (step1 SUCCEEDED)'
'step2 ERROR_CODE NOT IN (1,2,3)'
```

You can also refer to attributes of chain steps of the chain (this is called bind-variable syntax). The syntax is as follows:

```
STEP_NAME.ATTRIBUTE
```

- Possible attributes are: `completed`, `state`, `start_date`, `end_date`, `error_code`, and `duration`.
- Possible values for the state attribute include: `'NOT_STARTED'`, `'SCHEDULED'`, `'RUNNING'`, `'PAUSED'`, `'SUCCEEDED'`, `'FAILED'`, and `'STOPPED'`.
- If a step is in the state `'SUCCEEDED'`, `'FAILED'`, or `'STOPPED'`, its completed attribute is set to `'TRUE'`; otherwise, completed is `'FALSE'`.

Some examples of the bind variable syntax are:

```
':step1.state=''SUCCEEDED'' and
:step2.error_code=3'
'1=1'
':step3.state != ''COMPLETED'''
':step2.error_code not in (1,2,3)'
':step1.state = ''NOT_STARTED'''
```

The rule action specifies what is to be done as a result of the rule being triggered. A typical action is to run a specified step. Possible actions include:

- `START step_1[,step_2...]`
- `STOP step_1[,step_2...]`
- `END [{end_value | step_name.error_code}]`

When the job starts and at the end of each step, all rules are evaluated to see what action or actions occur next. You can also configure rules to be evaluated at regular intervals by using the EVALUATION_INTERVAL attribute of the chain.

You add a rule to a chain with the DEFINE_CHAIN_RULE procedure:

```
BEGIN
 DBMS_SCHEDULER.DEFINE_CHAIN_RULE (
   CHAIN_NAME => 'bulk_load_chain',
   CONDITION  => 'TRUE', -- starting step
   ACTION     => 'START load_data_evt,
                  stop_when_disk_full_evt',
   Rule_Name  => 'dataload_rule1',
   COMMENTS   => 'start the chain');

 DBMS_SCHEDULER.DEFINE_CHAIN_RULE (
   CHAIN_NAME => 'bulk_load_chain',
   CONDITION  => 'load_data_evt COMPLETED',
   ACTION     => 'START do_bulk_load',
   RULE_NAME  => 'dataload_rule2');
END;
```

4. Enable a chain with the ENABLE procedure (A chain is always created disabled). Enabling an already enabled chain does not return an error.

```
DBMS_SCHEDULER.ENABLE ('bulk_load_chain');
```

5. To run a chain, you must create a job of type 'CHAIN'. The job action must refer to the chain name.

```
BEGIN
  DBMS_SCHEDULER.CREATE_JOB (
    job_name        => 'bulk_load_chain_job',
    job_type        => 'CHAIN',
    job_action      => 'bulk_load_chain',
    repeat_interval => 'freq=daily;byhour=7',
    enabled         => TRUE);
END;
```

Managing Job Chains

- The RUN_CHAIN procedure immediately runs a chain by creating a run-once job with the job name given. If no job_name is given, one is automatically assigned in the form SCHED_CHAIN_JOB${N}. If a list of start steps is given, only those steps are started when the chain begins running (steps that would have normally started are not run). If no list of start steps is given, then the chain starts normally

  ```
  DBMS_SCHEDULER.RUN_CHAIN('chain_name','job_nam
  e','steps_to_start')
  ```

- The DROP_CHAIN_RULE procedure removes a rule from an existing chain. If dropping this rule makes the chain invalid, the user should first disable the chain to ensure that it does not run.

  ```
  DBMS_SCHEDULER.DROP_CHAIN_RULE('chain_name','r
  ule_name')
  ```

- Disable a chain

  ```
  DBMS_SCHEDULER.DISABLE('chain_name')
  ```

- Drop a chain

  ```
  DBMS_SCHEDULER.DROP_CHAIN('chain_name')
  ```

- Alter the SKIP or PAUSE attributes of a chain step by using the ALTER_CHAIN procedure. The ALTER_CHAIN procedure affects only future runs of the specified steps.

- Alter the steps in a running chain by using the ALTER_RUNNING_CHAIN procedure

- Drop a step from a chain by using the DROP_CHAIN_STEP procedure

Monitoring Job Chains

DBA_SCHEDULER_CHAINS
: contains information about the chain owner and name; the rule set name and rule set owner for the chain; the number of rules; the number of steps; whether or not the chain is enabled; whether or not the chain uses an evaluation interval; and whether or not the chain uses a user-defined rule set.

DBA_SCHEDULER_CHAIN_RULES
: displays the name and owner of the chain for which the rule was defined; the rule name, owner, and condition; and the action to be performed if the condition evaluates to TRUE.

DBA_SCHEDULER_CHAIN_STEPS
: displays the name and owner of the chain for which the step was created; the step name; the program name and owner; whether the step should be skipped or not; and whether or not the step should be paused after it completes.

DBA_SCHEDULER_RUNNING_CHAINS
: contains the chain name and owner; the name and owner of the job that points to the chain; the name of the steps in the chain and their current state; errors encountered by the chain step; the time at which the chain step started and ended; how long it took the step to complete; and the name of the job running the step, if it is current executing.

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_MAPPIN
G_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
(PLAN => 'SERVICE_LEVEL_PLAN',
 CPU_MTH -> 'RATIO',
 COMMENT => 'SERVICE LEVEL PLAN');

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

... and so on to other groups.

Switching Plans at Scheduler Window Boundaries

The Scheduler can automatically change the Resource Manager plan at the Scheduler window boundaries. In some cases, this may be unacceptable. For example, if you have an important task to finish, and if you set the Resource Manager plan to give your task priority, then you expect that the plan will remain the same until you change it.

To prevent Resource Manager plan to change while your task is running you have the following options:

- Set the RESOURCE_MANAGER_PLAN parameter to the name of the plan you want for the system and prefix the name with FORCE:. Using the prefix FORCE indicates that the current Resource Manager plan can be changed only when the database administrator changes the value of the RESOURCE_MANAGER_PLAN parameter. This restriction can be lifted by reexecuting the command without prefixing the plan name with FORCE:.
- Set the ALLOW_SCHEDULER_PLAN_SWITCHES flag to FALSE in the DBMS_RESOURCE_MANAGER.SWITCH_PLAN package procedure.
- Using Database Control, you can do this by deselecting the Automatic Plan Switching Enabled check box in the Resource Plan page.

Monitoring the Resource Manager

To manage and monitor the Resource Manager by using EM Database Control, on the **Administration** tabbed page | **Resource Manager** section | **Monitors** link.

Note: When you activate a plan by using the Resource Monitors page, you must exit the page and then choose Resource Monitors again to update the page and view the statistics for the newly activated plan.

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 you 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:

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

V$FLASH_RECOVERY_AREA_USAGE

This view provides information about the flash recovery area disk space usage. Following is its main columns:

FILE_TYPE	the type of the file and can have any of the following values: controlfile, onlinelog, archivelog, backuppiece, imagecopy, flashbacklog
PERCENT_SPACE_USED	This represents the disk space used by the file type, in percentage.
PERCENT_SPACE_RECLAIMABLE	this represents the percentage of disk space reclaimable from the file type after deleting any obsolete or redundant files, and files backed up to a tertiary device.

Flash Recovery Area Columns in Other Views

The Yes/No column IS_RECOVERY_DEST_FILE is added to some dictionary views to indicate whether the file was created in the flash recovery area. It exists in V$CONTROLFILE, V$LOGFILE, V$ARCHIVED_LOG, V$DATAFILE_COPY, V$BACKUP_PIECE.

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. This takes much less time than performing a full image copy of the datafiles every day.

This is applied through two phases:

1. Apply the incremental backups to datafile image copies. This is done at the database block level.
2. Then apply the archive logs since the last incremental backup only. This is done at the transaction level (slower than previous phase).

To implement this option, you do the following steps:

1. Use the BACKUP INCREMENTAL LEVEL 1...FOR RECOVER OF COPY WITH TAG ... form of the BACKUP command to create incremental backups that can be incrementally updated. If an incremental level 0 backup does not already exist, then executing this command creates a level 0 backup with the specified tag.
2. Apply any incremental backups to a set of data file copies with the same tag using the RECOVER COPY ...WITH TAG ... form of the BACKUP command. Tags must be used to identify the incremental backups and data file copies created for use in this strategy, so that they do not interfere with other backup strategies that you implement.

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;
```

Enhancements in RMAN

Configuration Changes

- When opening a database that was recovered using a backup or a current control file, the Oracle server automatically re-creates any locally managed temporary files if they do not exist. The files are re-created with the same creation size and storage settings. You can change the temporary file names by using the RMAN SET NEWNAME FOR TEMPFILE and SWITCH TEMPFILE command options.
- You no longer need to re-create the control file when any of the following configuration parameters are modified: MAXLOGFILES, MAXLOGMEMBERS, MAXINSTANCES. This ensures that you do not lose the RMAN catalog metadata through control file re-creation, and reduces system down-time requirements.
- RMAN backups to backup sets do not back up never-used blocks. In addition, Oracle Database 10g no longer backs up nonused blocks in locally managed tablespaces. Examples of such blocks are blocks that belong to dropped or truncated objects.

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 backuset):

```
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"
```

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
```

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.

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; }
```

Using the Database Control to Configure Backups

On the Database Control home page, follow the links: 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

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
```

Note: Consider this fast recovery method as a temporary solution. Later, you should relocate your database datafiles to permanent locations.

This method applies in the tablespace level as well:

```
RMAN> sql 'alter tablespace users offline';
RMAN> switch datafile 4 to copy;
RMAN> recover datafile 4;
RMAN> sql 'alter tablespace users online';
```

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 .. AS .. statement. Now, in Oracle Database 10g, you can create the lost file as part of an RMAN RESTORE DATABASE command.

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 does not 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.

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.

This automatically generated instance will be in the same database server. Remember, as with previous versions, instance creation introduces performance overhead during the recovery operation.

Automatic creation of Temporary Datafiles

Starting from release 2, Temporary datafiles that belong to locally managed temporary tablespaces are automatically re-created during RMAN recovery operation. This eliminates the need to manually create temporary tablespaces after recovery.

New RMAN Dynamic Performance Views

In Oracle 10.2 and above, in order to provide more details about its operation, RMAN is supported by a number of new dynamic performance views including:

```
V$BACKUP_ARCHIVELOG_DETAILS
V$BACKUP_ARCHIVELOG_SUMMARY
V$BACKUP_CONTROLFILE_DETAILS
V$BACKUP_CONTROLFILE_SUMMARY
V$BACKUP_COPY_DETAILS
V$BACKUP_COPY_SUMMARY
V$BACKUP_DATAFILE_DETAILS
V$BACKUP_DATAFILES_SUMMARY
V$BACKUP_JOB_DETAILS
V$BACKUP_PIECE_DETAILS
V$BACKUP_SET_DETAILS
V$BACKUP_SET_SUMMARY
V$BACKUP_SPFILE_DETAILS
V$BACKUP_SPFILE_SUMMARY
```

One other useful view is V$RMAN_BACKUP_JOB_DETAILS. It informs you about history of all backups done by the RMAN. You will find details like how long the backup took, how many RMAN jobs have been issued, the status of each job, what time they started and completed, rate of the backup produced and how fast data was read and written by the process.

```
COL STATUS FORMAT A9
COL HRS FORMAT 999.99
SELECT
 SESSION_KEY, INPUT_TYPE, STATUS,
 TO_CHAR(START_TIME,'DD/MM/YY HH24:MI')
START_TIME,
 TO_CHAR(END_TIME,'DD/MM/YY HH24:MI')
END_TIME,
 ELAPSED_SECONDS/3600   HRS
FROM V$RMAN_BACKUP_JOB_DETAILS
ORDER BY SESSION_KEY
```

```
COL INS FORMAT A10
COL OUTS FORMAT A10
SELECT SESSION_KEY,
 OPTIMIZED,
 COMPRESSION_RATIO,
 INPUT_BYTES_PER_SEC_DISPLAY INS,
 OUTPUT_BYTES_PER_SEC_DISPLAY OUTS,
 TIME_TAKEN_DISPLAY
FROM V$RMAN_BACKUP_JOB_DETAILS
ORDER BY SESSION_KEY
```

Another new view is V$RMAN_BACKUP_TYPE. It informs the type of backups performed by RMAN : BACKUPSET, SPFILE, CONTROLFILE, ARCHIVELOG, DATAFILE INCR, DATAFILE FULL, DB INCR, RECVR AREA and DB FULL.

The view V$RMAN_OUTPUT records the output (log) from the RMAN jobs so that you can view it later. This view is useful for checking log of scripted RMAN jobs as well as ad-hoc jobs.

Note: All those view are in-memory and cleared upon the instance shut down.

Oracle Secure Backup

- In Oracle Database 10g Release 2, a new tool called Oracle Secure Backup (OSB) is available. OSB enables you to use RMAN to backup directly to tape library without using the costly third-party MML layer.
- OSB can be controlled and administered via Oracle Enterprise Manager or obtool command-line.
- Beside backing up the database, OSB can be used to backup filesystems as well.

Cross-Platform Transportable Database

- The RMAN CONVERT DATABASE command is used to automate the movement of an entire database from one platform (the source platform) to another (the destination platform).
- The source and destination platform must share the same endian format.

Note: If the source and destination platform have different endian format, you can create a new database on a destination platform manually, and transport needed tablespaces from the source database using cross-platform transportable tablespace.

A. Preparing for CONVERT DATABASE: Using the DBMS_TDB Package

A.1 Using DBMS_TDB.CHECK_DB to Check Database State

DBMS_TDB.CHECK_DB checks whether a database can be transported to a desired destination platform, and whether the current state of the database permits transport.

It is a function that returns TRUE if the source database can be transported using CONVERT DATABASE, and FALSE otherwise.

It has the following parameters:

TARGET_PLATFORM_NAME
The name of the destination platform, as it appears in V$DB_TRANSPORTABLE_PLATFORM.

SKIP_OPTION
Specifies which, if any, parts of the database to skip when checking whether the database can be transported. Supported values (of type NUMBER) are:

- SKIP_NONE (or 0), which checks all tablespaces
- SKIP_OFFLINE (or 2), which skips checking datafiles in offline tablespaces
- SKIP_READONLY (or 3), which skips checking datafiles in read-only tablespaces

Set SERVEROUTPUT to ON to see output includes why the database cannot be transported.

```
SET SERVEROUTPUT ON
DECLARE
 DB_READY BOOLEAN;
BEGIN
 DB_READY := DBMS_TDB.CHECK_DB('Microsoft
Windows IA (32-bit)',DBMS_TDB.SKIP_READONLY);
END;
```

A.2 Using DBMS_TDB.CHECK_EXTERNAL to Identify External Objects

DBMS_TDB.CHECK_EXTERNAL must be used to identify any external tables, directories or BFILEs. RMAN cannot automate the transport of such files.

```
SET SERVEROUTPUT ON
DECLARE
     EXTERNAL BOOLEAN;
BEGIN
 /* value of external is ignored */
 EXTERNAL := DBMS_TDB.CHECK_EXTERNAL;
END;
```

B. Using the RMAN CONVERT DATABASE Command

1. Open the database in READ ONLY mode then use the RMAN command CONVERT DATABASE as follows:

```
CONVERT DATABASE
 NEW DATABASE 'newdb'
 TRANSPORT SCRIPT '/tmp/convertdb/transcript'
 TO PLATFORM 'Microsoft Windows IA (32-bit)'
 DB_FILE_NAME_CONVERT '/disk1/oracle/dbs'
'/tmp/convertdb'
```

Alternatively, you can use ON TARGET PLATFORM clause makes the datafile *conversion on the target* database. This means RMAN will generate the appropriate *conversion* scripts to be run on the target server to perform the datafile conversion there.

```
CONVERT DATABASE
 ON TARGET PLATFORM
 NEW DATABASE 'newdb'
 CONVERT SCRIPT '/tmp/convertdb/newdb.cnv'
 TRANSPORT SCRIPT '/tmp/convertdb/transcript'
 DB_FILE_NAME_CONVERT = '/disk1/oracle/dbs'
'/tmp/convertdb'
```

Note: CONVERT DATABASE ON TARGET PLATFORM does not produce converted datafile copies.

2. Copy all of the files produced to the destination host and place the datafiles in the desired locations on the destination host.
3. If the path to the datafiles is different on the destination, then edit, if any, the conversion script and the transport script to refer to the new datafile locations.

```
CONVERT DATAFILE
'/tmp/SYSTEM01.DBF'
 FROM PLATFORM 'Microsoft Windows IA (32-bit)'
 FORMAT
'/u01/app/oracle/oradata/newdb/system01.dbf'
```

4. Edit the PFILE to change any settings for the destination database.
5. Execute the transport script in SQL*Plus to create the new database on the destination host.

```
SQL> @transcript
```

6. Open the database using RESETLOGS option

```
ALTER DATABASE OPEN RESETLOGS;
```

7. Execute some maintenance code:

```
ALTER TABLESPACE TEMP
 ADD TEMPFILE
'/u01/app/oracle/oradata/newdb/temp01.tmp'
 SIZE 202375168 AUTOEXTEND ON NEXT 655360
 MAXSIZE 32767M

SHUTDOWN IMMEDIATE;

STARTUP UPGRADE
PFILE='/u01/../newdb/INIT_RPTREPOS.ORA'
SQL> @@ ?/rdbms/admin/utlirp.sql

SHUTDOWN IMMEDIATE;

SQL> STARTUP
PFILE='/u01/../newdb/INIT_RPTREPOS.ORA'

@@ ?/rdbms/admin/utlrp.sql
```

8. Change the internal database identifier using DBNEWID Utility

Restore Points

- A restore point serves as an alias for an SCN or a specific point in time.
- It is stored in the control file.
- It can be used with: RECOVER DATABASE, FLASHBACK DATABASE, FLASHBACK TABLE.
- It requires the database to operate in ARCHIVEMODE and the database flashback logging enabled.
- Use V$RESTORE_POINT to obtain information about restore points

```
CREATE RESTORE POINT before_load
```

Guaranteed Restore Points

- A guaranteed restore point ensures that you can perform a FLASHBACK DATABASE command to that SCN at any time.
- It can be used to restore beyond time specified by DB_FLASHBACK_RETENTION_TARGET parameter
- Even the effects of NOLOGGING operations such as direct load inserts can be reversed using guaranteed restore points.
- It must be manually deleted.

```
CREATE RESTORE POINT before_load GUARANTEE
FLASHBACK DATABASE
```

To flashback database to a restore point

```
SHUTDOWN IMMEDIATE
STARTUP MOUNT
FLASHBACK DATABASE TO RESTORE POINT before_load
ALTER DATABASE OPEN READ ONLY
-- check the flashed back data
SHUTDOWN IMMEDIATE
STARTUP MOUNT
ALTER DATABASE OPEN RESETLOGS;
```

To recover the database to the restore point

```
RECOVER DATABASE UNTIL RESTORE POINT
before_load
```

To flashback table to restore point

```
ALTER TABLE emp ENABLE ROW MOVEMENT
FLASHBACK TABLE emp TO RESTORE POINT
before_load
```

To drop a restore point

```
DROP RESTORE POINT before_load
```

Placing All Files in Online Backup Mode

- In Oracle Database 10g, with a single command you can place all the data files in the database in online backup mode. You no longer need to place each tablespace in online backup mode individually. This makes user-managed online backup easier. The requirements for implementing the user-managed online backup still apply.

```
ALTER DATABASE BEGIN BACKUP
```

- When you issue this command, any nonexistent, offline, or read-only data files are simply skipped and the processing continues.
- However, for the command `ALTER DATABASE END BACKUP`, if you have a data file with an OFFLINE file status, you receive a warning message. Also, with this command, if you have a tablespace in read-only mode, you do not receive any messages.

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 archivelog 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 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$ksisyyg0TxKnt18rqukpQA==$0"
TO BEFORE DROP RENAME TO NEW_PERSONS
```

Note: When you flashback a table, Oracle will recover the dependent objects (except bitmap indexes) as well, but they will 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 particualr 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/rdsk/*s1',
'/dev/rdsk/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

In Oracle 10g Release 1, 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.

In release 2, in DBCA you will see an option named as *Disk Group Management*. This option leads to ASM instance creation steps.

Creating the ASM Instance Manually (on Windows)

6. If CSS service is not there, create it by executing the following bat file:

```
<orahome>\bin\localconfig add
```

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

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

8. 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'
```

Note: The undocumented parameter _ASM_ALLOW_ONLY_RAW_DISKS is used to make the instance recognize virtual disks created in previous step. Of course, in a production database, this parameter is not used.

9. Create the ASM instance service:

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

10. 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 filenaming 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_ALIAS`. 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 files'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.

Database Instance Parameter Changes

Increase shared pool size based on the following guidelines:

- For disk groups using external redundancy: Every 100 GB of space needs 1 MB of extra shared pool plus a fixed amount of 2 MB of shared pool.
- For disk groups using normal redundancy: Every 50 GB of space needs 1 MB of extra shared pool plus a fixed amount of 4 MB of shared pool.
- For disk groups using high redundancy: Every 33 GB of space needs 1 MB of extra shared pool plus a fixed amount of 6 MB of shared pool.

To obtain the current database storage size that is either already on ASM or will be stored in ASM:

```
SELECT d+l+t DB_SPACE
FROM
 (SELECT SUM(bytes)/(1024*1024*1024) d
  FROM v$datafile),
 (SELECT SUM(bytes)/(1024*1024*1024) l
  FROM v$logfile a, v$log b
  WHERE a.group#=b.group#),
 (SELECT SUM(bytes)/(1024*1024*1024) t
  FROM v$tempfile
  WHERE status='ONLINE')
```

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.

```
DB_CREATE_FILE_DEST = '+dgroup1'
DB_RECOVERY_FILE_DEST = '+dgroup2'
```

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.ctl';
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

Dynamice 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.

ASM and Transportable Tablespaces

During the transportation of a tablespace from one database to another, it is possible for your source and target tablespaces to be stored either using ASM files or regular file-system files.

In all possible storage combinations, you can perform the transfer by using the DBMS_FILE_TRANSFER package running in one of the database instances. This operation can be performed directly without having to convert the data file.

For information about transportable tablespaces, refer to section "Transporting Tablespaces Across Platforms".

For information about using DBMS_FILE_TRANSFER, refer to section "Copying Files Using the Database Server".

ASM Command-Line Interface

Introduced in Oracle 10g release 2, the ASM Command-Line Interface (ASMCMD) utility provides an easy way to manipulate files within Automatic Storage Management (ASM) diskgroups. Its major functionality is to present an ASM file system in a user-friendly directory-tree structure. ASMCMD provides short commands for accessing the files and directories within ASM diskgroups.

The interface provides both interactive and noninteractive modes. The interactive mode enters a shell-like environment where the user is prompted to issue the commands. The noninteractive mode executes a single command and exits the utility. The latter is made available for scripting and batch-processing purposes.

You can invoke the ASMCMD tool with a -p parameter to always display the present directory inside the prompt itself.
Here is a brief description of ASMCMD commands:

pwd	displays the absolute path of the current directory.
cd	changes the current directory to the specify directory.
find	finds under a specified directory all paths that match a given pattern.
ls	lists aliases or its contents alphabetically by name if the alias is a directory.
mkdir	creates directories.
rm	removes the specified file as well as its system alias. If it is an empty

directory, then rm removes it.

mkalias	creates the specified user alias for the specified system alias.
rmalias	deletes the specified user aliases, while preserving the files and their system aliases.
du	displays the total space used for files located recursively under the specified directory.
lsdg	lists all diskgroups and their attributes.
lsct	lists all clients and their attributes.
help	displays list of commands

FTP and HTTP Access

Because ASM is not a regular file system, you can't use the standard FTP and HTTP services to access these files. To access them, you can use the file mapping functionalities provided by the Oracle XML Database (Oracle XML DB) feature.

To set up the FTP access, you must first set up the Oracle XML DB access to the ASM folders. I can do this by executing the `catxdbdbca.sql script`, found in the `$ORACLE_HOME/rdbms/admin` directory. The script takes two parameters: the port numbers for the FTP and HTTP services, respectively.

```
@catxdbdbca 7777 8080
```

Now you can connect to the created Oracle XML DB FTP service using a database username and password:

```
ftp myserverhost 7777
```

ASM disk groups are available outside the database via a virtual file system: `/sys/asm`. From there, you can navigate ASM storgae. For example:

```
ftp> cd /sys/asm
ftp> ls
USERDG5
USERDG4
USERDG3
USERDG2
USERDG1
ftp> cd USERDG2
250 CWD Command successful
ftp> ls
emrep
DBA102
ftp> cd DBA102
ftp> ls
DATAFILE
system01.dbf
system01.dbf
sysaux01.dbf
undotbs01.dbf
users01.dbf
CONTROLFILE
control01.ctl
...
```

You can then switch to binary mode and download any datafile:

```
ftp> bin
ftp> get users01.db
```

For HTTP access, open the browser on the following URL:

```
http://myserverhost:8080
```

The browser connects to Oracle XML DB via HTTP. Click on the hyperlink sys and then asm; you will then see all the disk groups from where you can download any datafile.

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,
      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` caluse 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)
```

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.

TASK_NAME	to identify and query a particular TUNE_MVIEW recommendation.
ACTION_ID	column shows the command order number.
SCRIPT_TYPE	CREATE, DROP, UNKNOWN
STATEMENT	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\utlxrw.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 INTEGER	Cost of original query
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 MATERIALIZE 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.

Database Security

In Oracle 10.2 auditing records can be output in XML. XML files are known to be portable and readable. Furthermore, they can easily be parsed by any XML parser to extract useful information from them.

```
ALTER SYSTEM SET audit_trail = 'XML' SCOPE =
SPFILE
```

```
The previous setting does not record the SQL
statement issued by the session. To do that you
should enable the extended XML auditing option
by issuing the following command:
```

```
ALTER SYSTEM SET audit_trail = 'XML, extended'
SCOPE = SPFILE
```

Audit trail files are written to directory specified by AUDIT_DUMP_DEST parameter

Defaults to $ORACLE_BASE/admin/<SID>/adump

Note: You can still view contents of the auditing XML files using SQL commands by querying the view V$XML_AUDIT_TRAIL. The only difference between data displayed in the view and in XML files is the column EXTENDED_TIMESTAMP which has the UTC-based format in the XML files whereas it is displayed on the local time zone in the view.

Following is a table of tags in the XML file:

Tag	Description
AUDIT_RECORD	Tag to delimit an audit entry
AUDIT_TYPE	possible values are: 1 standard XML audit 2 fine-grained audit 4 sys operation audit 8 mandatory XML audit
SESSION_ID	this is equivalent to AUDSID in v$session
STATEMENTID	each statement issued by the session has an id
EXTENDED_TIMESTAMP	audit recording UTC-based time
DB_USER	database username,
OS_USER	OS username
USERHOST	host name
OS_PROCESS	OS process ID
TERMINAL	connected terminal
INSTANCE_NUMBER	instance number in case of RAC
OBJECT_SCHEMA	owner of the object manipulated
OBJECT_NAME	name of the object
ACTION	action performed by the user
RETURNCODE	possible values are: 0 the statement succeeded n error number returned
SCN	SCN number
SESACTIONS	16 characters in which the first 12 could be of the following: - : action(s) not performed S : action(s) succeeded F : action(s) failed B : both (S and F) location of the characted indicates the action as follows: (1)Alter, (2)Audit, (3)Comment, (4)Delete, (5)Grant, (6)Index, (7)Insert, (8)Lock, (9)Rename, (10)Select, (11)Update, and (12)Flashback
SQL_BIND	values of bin variables displayed in Sql_Text tag.
SQL_TEXT	SQL statment issued by the session. Available only when Exteneded XML auditing is enabled.

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.

Oracle Transparent Data Encryption (TDE)

- TDE is an automatic mechanism encryption of sensitive information. There is no need to change application logic. It encrypts data and index values on the *disk*.
- It uses an opened Oracle Wallet to generate a master key for the entire database.
- Column length changes on disk.
- Actual lengths not reported by DUMP or VSIZE.

Setting up TDE

1. Create the Wallet file:

add the following to the sqlnet.ora

```
ENCRYPTION_WALLET_LOCATION =
 (SOURCE=
   (METHOD=file)
     (METHOD_DATA=
(DIRECTORY=C:\oracle\OraDb10g\admin\ora10g\wall
et)))
```

Note: Make sure the indicated folder exists.

Note: Alternatively, you can use Oracle Wallet Manager.

Caution: Wallet file must be included in your backup.

2. Set the master key

This is done only ***once***:

```
ALTER SYSTEM SET ENCRYPTION KEY IDENTIFIED BY
<password>;
```

3. Create tables that contain encrypted columns

```
CREATE TABLE emp (
 first_name VARCHAR2(128),
 ...
 empID NUMBER ENCRYPT NO SALT,
 salary NUMBER(6) ENCRYPT USING '3DES168',
 comm NUMBER(6) ENCRYPT);
ALTER TABLE EMP MODIFY ( SAL ENCRYPT NO SALT )
```

Note: possible algorithms are AES128, (AES192), AES256, or 3DES168

Note: the *salt* increases the protection but prevents indexing on the column.

Existing Tables and TDE

Add encrypted columns:

```
ALTER TABLE emp ADD (ssn VARCHAR2(11) ENCRYPT);
```

Encrypt unencrypted columns:

```
ALTER TABLE emp MODIFY (first_name ENCRYPT);
```

Disable column encryption:

```
ALTER TABLE emp MODIFY (first_name DECRYPT);
```

Add or remove salt:

```
ALTER TABLE emp MODIFY (first_name ENCRYPT [NO]
SALT);
```

Change keys and the encryption algorithm:

```
ALTER TABLE emp REKEY USING '3DES168';
```

To Test TDE

```
SELECT
DBMS_ROWID.ROWID_TO_ABSOLUTE_FNO
(ROWID,USER,'EMP'),
DBMS_ROWID.ROWID_BLOCK_NUMBER (ROWID) FROM EMP;
File Nu Block Number
------- ------------
4       63
ALTER SESSION SET EVENTS  '10389 trace name
context forever, level 1';
ALTER SYSTEM DUMP DATAFILE 4 BLOCK 63;
```

Opening and Closing the Wallet

The Wallet must be opened after instance restart.

```
ALTER SYSTEM SET ENCRYPTION WALLET OPEN
IDENTIFIED BY password>
ALTER SYSTEM SET ENCRYPTION WALLET CLOSE
```

TDE and Data Pump Export and Import

Use your own provided column key during export and import:

```
expdp hr/hrpswd directory=dir_dp tables=emp
ENCRYPTION_PASSWORD=testme
impdp hr/hrpswd directory=dir_dp
ENCRYPTION_PASSWORD=testme
table_exists_action=replace tables=emp
```

RMAN Encrypted Backups

Three possible encryption modes for your backups:

- Transparent mode: It requires Oracle Wallet. It is best suited for day-to-day backup and restore operations at the same location. It is the default encryption mode.

```
CONFIGURE ENCRYPTION FOR DATABASE ON
```

- Password mode: It requires you to provide a password. It is best suited for backups restored at remote locations.

```
SET ENCRYPTION ON IDENTIFIED BY password ONLY
```

- Dual mode: It can use either Oracle Wallets or passwords.

After making sure the wallet is open,

```
SET ENCRYPTION ON IDENTIFIED BY password
```

If there is no wallet or the wallet is closed:

```
SET DECRYPTION IDENTIFIED BY password1 {,
password2,…, passwordn}
```

Secure External Password Store

- Username and password credentials for connecting to databases can now be stored in a client-side Oracle wallet, a secure software container used to store authentication and signing credentials.
- When this feature is configured, application code, batch jobs, and scripts no longer need embedded user names and passwords.

Note: You cannot use Oracle Wallet Manager to manage credentials in external password store of the wallet. Instead, you can use the command-line utility `mkstore`.

The username and password for the following command are obtained from the wallet. The autologin feature of this wallet is turned on so the system does not need a password to open the wallet.

```
connect /@db_connect_string
```

To enable clients to use the external password store:

1. Create an autologin wallet on the client by using the following syntax at the command line:

```
mkstore -wrl <wallet_location> -create
```

2. Create database connection credentials in the wallet by using the following syntax at the command line:

```
mkstore -wrl <wallet_location> -
createCredential <db_connect_string> <username>
<password>
```

3. In the client `sqlnet.ora` file:

```
WALLET_LOCATION =
   (SOURCE =
     (METHOD = FILE)
     (METHOD_DATA =
       (DIRECTORY = <wallet_location>)
     )
    )
```

4. To override external authentication, such as Windows native authentication or Secure Sockets Layer (SSL), in the client sqlnet.ora file:

```
SQLNET.WALLET_OVERRIDE = TRUE
```

Managing External Password Store Credentials

To list the contents of the external password store:

```
mkstore -wrl <wallet_location> -listCredential
```

To add database login credentials to an existing client wallet:

```
mkstore -wrl <wallet_location> -
createCredential <db_alias> <username>
<password>
```

To modify database login credentials in a wallet:

```
mkstore -wrl <wallet_location> -
modifyCredential <dbase_alias> <username>
<password>
```

To delete database login credentials from a wallet:

```
mkstore -wrl <wallet_location> -
deleteCredential <db_alias>
```

Connect Role Privilege Reduction

The `connect` role privilege reduction feature reduces the number of privileges granted to the connect role to one, the `CREATE SESSION` privilege.

Miscellaneous New Features

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>)
```

For example:

```
DBMS_MONITOR.SERV_MOD_ACT_STAT_ENABLE(
service_name=>'APPS1',module_name =>'PAYROLL')
```

To enable tracing for a Service named APPS1:

```
DBMS_MONITOR.SERV_MOD_ACT_TRACE_ENABLE('APPS1',
DBMS_MONITOR.ALL_MODULES,
DBMS_MONITOR.ALL_ACTIONS,TRUE,FALSE,NULL)
```

To disable tracing specified in the previous step:

```
DBMS_MONITOR.SERV_MOD_ACT_TRACE_DISABLE('APPS1'
)
```

Note: The `DBMS_APPLICATION_INFO` has two procedures, `SET_MODULE` and `SET_ACTION`, which allow programmers to specify module and action names.

Of course, statistic accumulation for a session ID is still possible:

```
DBMS_MONITOR.SESSION_TRACE_ENABLE
(SESSION_ID=>139, SERIAL_NUM=>53, WAITS=>TRUE,
 BINDS=>FALSE);
```

For information about tracing in the instance and database level, refer to "Database and Instance Level Trace".

Marking the Trace Files

You can also add your own marker to the trace file names so you can more easily find the generated files.

```
ALTER SESSION SET TRACEFILE_IDENTIFIER
="hr_report"
```

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.

```
trcsess output="hr_report.trc" service="APPS1"
module="PAYROLL" action="bulk load"
```

You can then run `TKPROF` against the consolidated trace file to generate a report.

```
..\udump> tkprof hr_report.trc
output=hr_trc_report SORT=(EXEELA, PRSELA,
FCHELA)
```

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 `prvtmail.plb` scripts located in your `ORACLE_HOME/rdbms/admin` directory.

Regular Expressions

Oracle provides the following regular expression functions for text complex searching:

- `REGEXP_LIKE`
- `REGEXP_REPLACE`
- `REGEXP_INSTRING`
- `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 `q` to provide your own quotation mark delimiters.

Enhancements in SQL* Plus

Easy Prompt Modification

In SQL*Plus 10.1.0.2, you can include the database username and what the user is connected as in the prompt using the following command:

```
set sqlprompt "_user _privilege>"
```

Date can also be displayed:

```
set sqlprompt "_user _privilege 'on' _date >"
```

Note: the date will be displayed in a format based on the active `NLS_DATE_FORMAT` value

Database identified can also be added:

```
set sqlprompt "_user'@'_connect_identifier>"
```

Enhancements in Spool Command

In 10g, the spool command can append to an existing one:

```
spool myspoolfile.lst append
```

If you want to overwrite it, simply omit the append clause or use `REPLACE` instead, which is the default.

```
spool myspoolfile.lst [replace]
```

The following will check the existence of the file before writing to prevent the overwriting:

```
spool myspoolfile.lst create
```

Executing Login.sql File

In Oracle Database 10g, the file `login.sql` is not only executed at SQL*Plus startup time, but at connect time as well. Therefore, each time you successfully issue the connect command, the `login.sql` script will be executed from the current directory.

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:

```
ALTER SESSION ENABLE RESUMABLE
```

Faster Startup

In Oracle Database 10g Release 2, when you start the instance, only 10% of the buffer cache is initialized; the rest is initialized after the database is opened by the checkpoint process. This new approach reduces instance startup time significantly.

Bear in mind, however, that until the entire buffer cache is initialized, automatic buffer cache sizing is not available.

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.

The `V$PROCESS_MEMORY` view

The `V$PROCESS_MEMORY` introduced in Oracle 10.2. It can be used to verify size of SQL and PL/SQL areas for a process. It is also included in STATSPACK report.

```
SELECT CATEGORY, ALLOCATED, USED, MAX ALLOCATED
FROM V$PROCESS_MEMORY WHERE pid = 26
```

Block Integrity Checking in Memory

Oracle ensures the data block's integrity by computing a checksum on the data value before writing the data block to the disk. This checksum value is also written to the disk. When the block is read from the disk, the reading process calculates the checksum again and then compares against the stored value. If they differ, it means the block is corrupted.

In Oracle Database 10g Release 2, you can make the database perform the check in memory as well (not only in disk). This is done by setting the initialization parameter `DB_BLOCK_CHECKSUM` to FULL.

Catching corruption in the memory will prevent it at the disk level as well as its propagation to the standby database.

Note: This option is by default disabled because the parameter `DB_BLOCK_CHECKSUM` has a default value of FALSE. Enabling this option introduce slight performance overhead.

`V$SESSION` Changes

The `V$SESSION` view enhanced to include tracing information of current session.

Three new columns now show the status of tracing:

- `sql_trace`—Shows (TRUE/FALSE) if SQL tracing has been enabled in the session
- `sql_trace_waits`—If session tracing is enabled, you can have the trace write wait information to the trace file; very useful in diagnosing performance issues.
- `sql_trace_binds`—If the session uses bind variables, you can have the trace write the bind variable values to the trace file. This column shows TRUE/FALSE.

```
BEGIN
 DBMS_MONITOR.SESSION_TRACE_ENABLE (
 SESSION_ID => 196,
 SERIAL_NUM => 60960,
 WAITS => TRUE,
 BINDS => FALSE );
END;
```

Note that the view V$SESSION is populated only if the procedure session_trace_enable in the package dbms_monitor is used to enable tracing, not by alter session set sql_trace=true or setting the event 10046.

The DBMS_OUTPUT package

DBMS_OUTPUT maximum line length

- In Oracle 10.1 and below - 255 bytes
- In Oracle 10.2 and above - 32767 bytes

DBMS_OUTPUT maximum output buffer size

- In Oracle 10.1 and below - 1000000 bytes
- In Oracle 10.2 and above - unlimited

The V$PARAMETER_VALID_VALUES view

The V$PARAMETER_VALID_VALUES view is introduced in Oracle 10.2. It returns one row for each valid value for each parameter taking scalar value.

```
SELECT name, value, isdefault FROM
v$parameter_valid_values WHERE name =
'cursor_sharing' ORDER BY ordinal
```

```
Parameter Name Value   IsDefault?
-------------- -----   ---------
cursor_sharing FORCE   FALSE
cursor_sharing EXACT   TRUE
cursor_sharing SIMILAR FALSE
```

Unicode 4.0

Oracle's Unicode character sets, AL32UTF8 and AL16UTF16, have been updated to support Unicode 4.0 in Oracle Database 10g Release 2. Unicode 4.0 has 1,226 new characters.