Exam
Manual

LearnSmart

(1Z0-040)

Oracle 10g

Oracle Certified Professional Upgrade

This LearnSmart exam manual prepares IT professionals for the Oracle 10g OCP upgrade certification exam. By studying this manual, exam candidates will spearhead their ability to display competence in an array of areas, including:

- Installation and Database Upgrades
- Automated Storage Management (ASM)
- Flash Recovery and RMAN Improvements
- SQL Enhancements
- And more!

Pave the way for success by purchasing this exam manual today!

Oracle 10g (1Z0-040) LearnSmart Exam Manual

Production Date: July 13, 2011
Product ID: 010276

Warning and Disclaimer

Every effort has been made to make this document as complete and as accurate as possible, but no warranty or fitness is implied. The publisher and authors assume no responsibility for errors or omissions. The information provided is on an "as is" basis. The authors and the publisher shall have neither liability nor responsibility to any person or entity with respect to any loss or damages arising from the information contained in this document.

Volume, Corporate, and Educational Sales

Favorable discounts are offered on all products when ordered in quantity. For more information, please contact us directly:

1-800-418-6789
solutions@learnsmartsystems.com

International Contact Information

International: +1 (813) 769-0920

United Kingdom: (0) 20 8816 8036

Table of Contents

Abstract

This Exam Manual prepares you for the Oracle 10g certification exam #1Z0-040, "Oracle 10g: New Features for Administrators." This test is popularly referred to as the "10g Upgrade Test." Passing it upgrades your Oracle 9i Certified Professional status to be current for Oracle 10g.

Topics on this exam include: installation, upgrading and configuration of 10g databases; loading and unloading data with Data Pump; new automated management features; performance management and problem analysis features; automating tasks with the new Scheduler; space management and data access improvements; improved support for very large databases; new features for recovery from logical errors; automatic storage management; SQL improvements; backup and recovery enhancements; security improvements; and many miscellaneous improvements.

What to Know

Understand how to upgrade existing Oracle databases to 10g, how to configure 10g databases, and how to configure servers that run 10g. Be aware of which initialization parameters are new and which have become obsolete by the 10g.

Know about the new Data Pump (the replacement for the traditional Export and Import utilities). Understand its architecture and how to use it with cross-platform transportable tablespaces, external tables, importing and exporting data and meta-data, and setting up and monitoring Data Pump jobs.

Understand 10g's new automatic management features. These include: automatic statistics collection, the Automatic Database Diagnostic Monitor (ADDM), Automatic Shared Memory Management (ASMM), the Automatic Workload Repository (AWR), and automatic undo retention tuning.

This test covers the many new "advisors" in 10g: the SQL Access Advisor, the SQL Tuning Advisor, ADDM as an automated problem-diagnosis tool, the Segment Advisor, the Undo Advisor, the PGA Advisor, the Buffer Cache Advisor, and the Redo Logfile Size Advisor. The exam expects you to be able to work with these automated facilities and Advisors either through 10g's new Graphical User Interface (GUI) panels or through the command line (presented by products like SQL*Plus). Know their default behaviors and how to alter them and use them through both interfaces. Two of the most important Advisors to understand and know how to use are the SQL Access Advisor and the SQL Tuning Advisor.

Know how to use the new Scheduler. Understand how to set up, run, manage, and monitor jobs. Know the roles of jobs, programs, schedules, and windows. Know how to reuse scheduler components and how to view information about job executions and job instances.

Tips

- You must know how to accomplish tasks both through the command line and via 10g's new Graphical User Interface (called the Enterprise Manager or EM). The test emphasizes the command line more than the GUI.
- You'll face lots of questions on package, procedure, and attribute names, as well as the data dictionary and V$ table names (the test designers apparently figure that if you know this stuff, you know how to use the features they underlie).
- General opinion has it that this is the most difficult Oracle Upgrade Test yet. There are more multiple-answer questions than before, and these are harder than single-answer questions. Also, the percentage required to pass (about 73%) is slightly higher than the 66-70% required by earlier Upgrade Exams.

To pass any Oracle exam, you need to do four things –

1. Verify Oracle Corp's exam requirements
2. Get hands-on experience
3. Study this Exam Manual, written specifically to help candidates pass the test
4. Work with practice questions

Verify Oracle Corp's Exam Requirements:
To verify Oracle Corp's exam requirements, go to their certification home page at www.oracle.com/education/certification, or go directly to the page for this test at http://education.oracle.com/pls/web_prod-plqdad/db_pages.getpage?page_id=44.

You need to verify the exam requirements because Oracle reserves the right to change them at any time (usually they only do so when a new release comes out, but if you're going to put in the effort to pass this test you'll want to make sure). The web site also lists the exact topics on the test, information about how long the test takes and how many questions it contains, and where and how to sign up to take the test.

Hands-on Experience:
To get hands-on experience with Oracle 10g, either get it at work or take advantage of Oracle's free 10g Database download. Go to http://www.oracle.com/database/index.html to get the free download, or go to www.oracle.com and enter "free download" in the Search Box. You can install the product on your Windows or Linux PC. The free license will give you what you need to pass the test.

Installation and Database Upgrades

New Installs

Install Oracle into a new, empty directory (not an existing Oracle directory).

If you choose to install them, companion products like the JPublisher, Legato Single Server Version (LSSV), Java Libraries, JAccelerator, Oracle InterMedia, and Oracle Text Knowledge Base install into the same directory as the Oracle Database. Oracle 10g HTTP Server (OHS) and HTML DB install into a different directory than Oracle Database. OHS and HTML DB must be installed into the same directory as one another.

Older-release clients will connect to 10g, but only 10g clients get all the new 10g features.

New installs require a minimum of 256M — or 512M with Database Control, the new 10g database management tool (described further below). Swap space should be 1G or two times the amount of server memory (RAM), and the required disk space includes about 1.5G for the 10g database and at least 1G of disk space for creating a database during installation.

Use the new Oracle Universal Installer (OUI), the GUI for installing 10g. Start it by running the **runInstaller** script under Unix/Linux. For Windows, just load the CD-ROM and let Window's AutoRun start the Installer. For Unix/Linux only, you must configure some kernel parameters before installing. Specify the Installer's **-ignoreSysPrereqs** option when starting the Installer if you want or need to ignore the checking of certain system prerequisites during installation.

The database installs from one CD. (Other CDs include: the Companion CD, 10g Client, Cluster Ready Services, and the Documentation Library.) The Oracle 10g Client is on a separate CD from the database. Installing the database software automatically installs a Client, but if you want only to install the Client, you must use the separate Client CD.

During the installation process, you'll be asked:

- Whether to create a starter database or not: If yes, you can choose either a *General Purpose*, *Transactional*, *Data Warehouse*, or *Advanced* database. If yes, the OUI later invokes the Database Configuration Assistant (DBCA) GUI tool to help you create the new database.
- Whether to use Grid Control or Database Control for managing the database: Both Grid Control and Database Control are components of the 10g web browser-based GUI management tool called the Enterprise Manager (EM). EM is the 10g replacement for the old GUI management tool, the Oracle Enterprise Manager (OEM). Grid Control provides central management of more than one database through the web browser-based EM GUI, while Database Control manages only one database.
- You may specify that OUI sets up automatic backups for the database. This sets up and schedules Recovery Manager (RMAN) backups. Default backup job execution time is 2 am with the target being the Flash Recovery Area of default size 2G (discussed later in this Exam Manual).
- The Database Configuration Assistant or DBCA asks whether to use file system files, raw devices, or Automatic Storage Management (ASM) for database storage. ASM is a new feature in 10g that automates storage management (discussed later in this Exam Manual). You will be familiar with filesystem and raw device-based databases from previous Oracle releases.

- DBCA installs 5 **sample schemas**: HR, IX, OE, PM, SH. These are different from the **database examples**, which are on the Companion install CD.
- DBCA performs many key database configuration tasks. These include configuring ASM, creating the new required **SYSAUX** tablespace, making the database ready for Grid or Database Control, creating management repository and services, implementing automated backups, and performing other configuration tasks.

Upgrading Existing Databases to 10g

Ways to upgrade existing Oracle databases to 10g:

- Direct
- Indirect
- Manually (via scripts)
- Via Import/Export utilities
- Copy data (via tools like the SQL*Plus **COPY** command **or CREATE TABLE AS SELECT**)

Direct upgrades are performed through the Database Upgrade Assistant (DBUA) GUI utility. The **only** releases you can directly upgrade are **8.0.6, 8.1.7, 9.0.1,** and **9.2.0**.

Indirect upgrades mean upgrading a database to one of these releases: 8.0.6, 8.1.7, 9.0.1, 9.2.0, and then using the direct upgrade methodology.

Manual upgrades mean running a set of scripts and performing certain steps manually. (If you use the DBUA GUI tool, it does these steps for you). Here are the steps:

1. Run **utlu101i.sql** script for advice on any required pre-install actions.
2. Minimum 10g Redo Log size is **10M**. Resize if necessary.
3. Do a cold database backup.
4. **SHUTDOWN NORMAL** the database (or **SHUTDOWN IMMEDIATE**).
5. Copy over parameter and password files. Update the database initialization parameter file (some pre-10g parameters are obsolete, others must be changed for 10g). Set **COMPATIBLE** parm to its 10g minimum required value of **9.2.0**. If you set this to 10.0, you cannot downgrade the database later. If using Windows, remove any Oracle instances using oradim. Then, using the same utility, create a new instance with the new parameter file.
6. **STARTUP UPGRADE** the database.
7. Create the new, required **SYSAUX** tablespace. This is a companion tablespace to **SYSTEM**. Both are required, cannot be renamed, and together comprise the minimal required tablespaces for a 10g database. SYSAUX must be: **PERMANENT**, **ONLINE**, **EXTENT MANAGEMENT LOCAL**, **SEGMENT SPACE MANAGEMENT AUTO**, and read/write. Its minimum recommended size is 500M.
8. Run the upgrade script for one of the four directly upgradeable releases:

   = u0800060.sql, 8.1.7 = u0801070.sql,
   9.0.1 = u0900010.sql, 9.2.0 = u0902000.sql
   Compile invalid DB objects using @?/rdbms/admin/utlrp.sql script.
9. Run the "upgrade Status" script **utlu101s.sql TEXT**.
10. Do a **SHUTDOWN IMMEDIATE**.

11. Update **listener.ora** and backup the database.
12. Start the database normally.

Databases can only be downgraded to 9.2.0, and then only if the COMPATIBLE parm is 9.2.0. Use STARTUP DOWNGRADE and run script d0902000.sql to downgrade. If you have ever set COMPATIBLE to 10.0 or above, you cannot downgrade the database!

Important — script **utlu101i.sql** runs before the upgrade to advise on pre-install actions you must take. It does not run any of those actions — you must do that — it only advises you. Script **utlu101s.sql** runs after the upgrade to show upgrade status and tell about any invalid component upgrades.

View **DBA_SERVER_REGISTRY** contains status info for 10g upgrades. Oracle uses the package **DBMS_REGISTRY** to determine the objects it will upgrade.

More on Configuration

10g initialization parameters are divided into **basic** and **advanced**. The 25 to 30 basic parameters are all that are needed for typical database configuration.

10g configures via policies, which categorize into object, storage, security, and configuration. Policy rules are prioritized as **High**, **Medium**, or **Informational**. Like most aspects of database configuration, policies can be managed via the Database Control GUI.

Nearly all aspects of 10g can be managed either by:

- Commands issued through the command line, plus manually interrogating the data dictionary and V$ views. (Tools like SQL*Plus and *i*SQL*Plus provide a command line.)
- Grid Control / Database Control (the Enterprise Manager or EM).

Two good examples of this are the new 10g features:

- Database Feature Usage – View **DBA_FEATURE_USAGE_STATISTICS** shows which new features of 10g are being used and how often. (Or see this information from EM.)
- High Water Mark (HWM) for Database Attributes – This shows high water mark statistics for a variety of database attributes (e.g., segment_size, db_size, number of datafiles, number of tablespaces, etc). View **DBA_HIGH_WATER_MARK_STATISTICS** tracks this information. (Or see it through the EM.)

The new **SYSAUX** tablespace is an auxiliary to the **SYSTEM** tablespace. Query view **V$SYSAUX_OCCUPANTS** to see which Oracle components use it to store their data. The column **MOVE_PROCEDURE** in that view indicates the procedure you execute to move the related occupant out of **SYSAUX** into another tablespace. As the view indicates, not all occupants can be moved out — for example, **STREAMS** cannot be moved out of **SYSAUX**. The view also shows space usage for each occupant. This example query shows the space usage of individual components in **SYSAUX**, as well as what the name of the procedures are that you would use to relocate occupants:

```
SELECT occupant_name, schema_name, space_usage_kbytes, move_procedure
FROM  V$SYSAUX_OCCUPANTS ;
```

opatch is a new 10g utility for applying one-off patches. The new GUI Enterprise Manager (EM) automatically checks the Oracle MetaLink website for new patches once daily. Oracle MetaLink is the Oracle Corporation web site from which you download patches and release upgrades.

It was mentioned earlier that you can choose between either (1) Grid Control or (2) Database Control when installing a 10g database. Grid Control provides central management of more than one database through the web browser-based EM GUI. It also manages other kinds of resources (called managed targets) like web servers, application servers, etc.

Grid Control requires one Oracle Management Agent (OMA) running on each server you will manage. On the central management server(s), it requires one Oracle Management Service (OMS), one Oracle Management Repository (OMR) for OMS to store information in, and one Grid Control Agent.

Figure 1: Grid Control Architecture

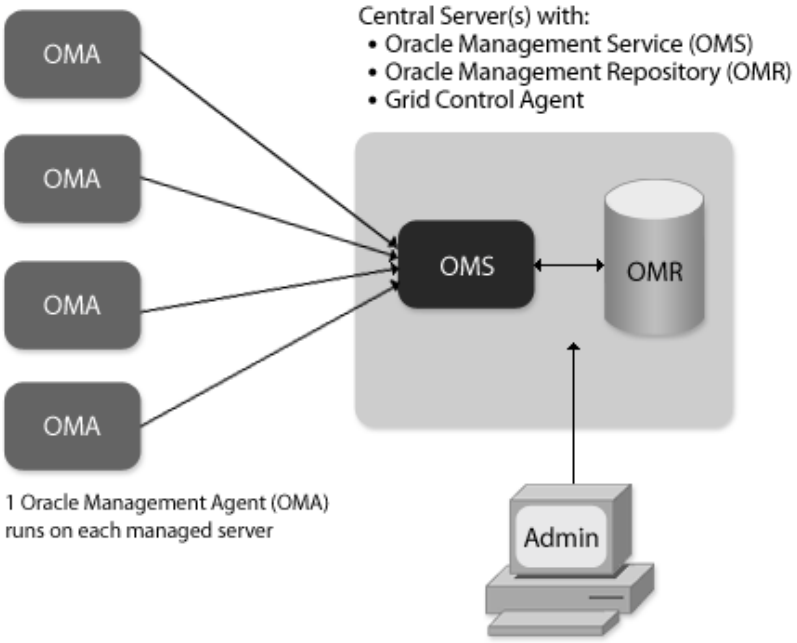

Figure 1 conceptually sketches the architecture through which Grid Control collects information on the resources it manages.

Database Control is the Enterprise Manager alternative to Grid Control. Database Control manages a single database through EM. It is installed by default and uses port 5500 on the database server. It can be viewed as a subset of the functionality of Grid Control -- it controls only a single database and does not support other kinds of managed targets. Access the Database Control via http://localhost:5500/em.

Cloning Databases

Cloning databases is a lot easier with 10g. 10g offers two entirely different approaches to cloning a database. Both are based on easy-to-use GUIs:

- Database Configuration Assistant (DBCA)
- EM "Clone Database" panels that use the Recovery Manager (RMAN)

DBCA – start DBCA via the **dbca** command or off the Windows Programs menu. DBCA is a GUI that creates databases from templates, which are XML files. There are two kinds of templates:

- Seed — This includes datafiles
- Non-seed — This is structure-only (no data)

DBCA puts the source database in the **MOUNT** state to create the template. After the template has been created, the database can be re-opened (the database is **OPEN** and available while the clone database is being created).

Two DBCA alternatives:

Type:	Use:	File Extensions:
Seed	Includes Data	**.dbc** for the template and **.dfj** for the data
Non-Seed	Structure only	**.dbt** for the template

EM "Clone Database" Option using RMAN – The Enterprise Manager panels "Clone Database" option takes a totally different approach than DBCA. It only clones databases installed via an OUI that has cloning support (the 10g OUI), and uses the EM job system and RMAN to achieve cloning. You can clone a single source database into multiple targets in a single cloning job. If the source database is in Archivelog mode, it remains up and running throughout the entire cloning operation.

EM Clone Database performs these steps for you:

1. Backs up database files
2. Transfers backup files from source to target(s)
3. Restores backup files to the target directories
4. Recovers the cloned database with archive logs
5. Opens the cloned database with the **RESETLOGS** option

The New Scheduler
A new scheduler based on package **DBMS_SCHEDULER** supersedes the old **DBMS_JOB** package (which is still present in 10g). Use the new package by coding or through the EM.

DBMS_SCHEDULER offers more power than the older **DBMS_JOB** package and it has these advantages:

- Executes 3 types of jobs (as specified by the **JOB_TYPE** parameter):
  - **EXECUTABLE** (executable scripts)
  - **STORED_PROCEDURE** (database stored procedures)
  - **PLSQL_BLOCK** (anonymous blocks)
- Is based on these main basic components:
  - **Job**: the program to run
  - **Program**: metadata about what to run (program name, type, arguments)
  - **Schedule**: when and how often to run the job
- The next 3 are the advanced components:
  - **Window**: activates different resource plans at different times; an interval of time associated with a specific resource plan
  - **Window Group**: a group of Windows
  - **Job Class**: a grouping of Jobs with the same resource requirements

Jobs, Programs, and Schedules are created in **user schemas**.

Job Classes, Windows, and Window Groups are created in the *SYS* schema.

These are key **DBMS_SCHEDULER** procedures through which you manage the above components:

CREATE_JOB	Creates a Job
CREATE_PROGRAM	Creates a Program
CREATE_SCHEDULE	Creates a Schedule
DEFINE_PROGRAM_ARGUMENT	Defines an argument for a Program
SET_JOB_ARGUMENT_VALUE	Sets parameters for a Job
CREATE_JOB_CLASS	Creates a Job Class
SET_ATTRIBUTE	Changes arguments of a scheduler component
SET_ATTRIBUTE_NULL	Sets an attribute to null (unsets an attribute)
ENABLE	Enables a Job or Program ***By default the ENABLED attribute is set to FALSE for Jobs and Programs***
DISABLE	Disables a Job or Program
RUN_JOB	Runs a Job

STOP_JOB	Stops a Job
DROP_JOB	Drops a Job from the Scheduler
RESET_JOB	Clears values for Job arguments
COPY_JOB	Copies all the attributes of an existing Job to a new Job
DROP_PROGRAM	Drops a Program from the Scheduler
DROP_PROGRAM_ARGUMENT	Drops arguments of a Program
DROP_SCHEDULE	Drops a Schedule
PURGE_LOG	Deletes the Scheduler's log files
CREATE_WINDOW	Creates a Window
DROP_WINDOWS	Drops a Window

JOBS — The Job component is the only absolutely required component to run a task. A Job can belong to only one Job Class. If the Job Class is not specified, the Job is assigned to the **DEFAULT_JOB_CLASS**. A Job's **JOB_PRIORITY** attribute can be set from **1** to **5** (1 is highest priority, 5 is lowest priority).

The following shows example code to create a Job:

```
EXEC   SYS.DBMS_SCHEDULER.CREATE_JOB(
job_name          => 'SCOTT.BACKUP',
job_type          => 'EXECUTABLE',
                          -- or PLSQL_BLOCK or STORED_PROCEDURE –
job_action        => '/scotts_scripts/run_backup.sh',
start_date        => to_date('08-03-2005 20:00:00','mm-dd-yyyy hh24:mi:ss'),
repeat_interval   => 'TRUNC(SYSDATE) + 23/24',
comments          => 'Scotts backup job',
enabled           => TRUE);      -- the default for ENABLED is FALSE –
```

The **REPEAT_INTERVAL** attribute specifies how often a Job or Schedule repeats. You can use a Calendaring Expression to specify it. There is always more than one way to specify the Job or Schedule **REPEAT_INTERVAL** when using Calendaring Expressions.

Calendaring Expressions have up to 3 parts:

Component:	Keyword:	Values:
Frequency*	FREQ	YEARLY, MONTHLY, WEEKLY, DAILY, HOURLY, MINUTELY, SECONDLY
Interval	INTERVAL	An integer between 1 & 99 inclusive
Specifier	BYMONTH, BYWEEKNO, BYYEARDAY, BYMONTHDAY, BYDAY, BYHOUR, BYMINUTE, BYSECOND	Allowable values depend on the Keyword you use

**Frequency* is the only mandatory part of a Calendaring Expression, though you would typically qualify it with either interval or specifier.

Examples:

Calendaring Expression:	Meaning:
FREQ=HOURLY; INTERVAL=8	Every 8 hours
FREQ=YEARLY;BYWEEKNO=4;BYDAY=SUN	The Sunday in the 4th week of the year
FREQ=HOURLY;BYMONTHDAY=1,-1	Hourly on the 1st and last days of the month

This example code creates a Schedule and uses a calendaring expression in its **REPEAT_INTERVAL**:

```
DBMS_SCHEDULER.CREATE_SCHEDULE(
repeat_interval          => 'FREQ=WEEKLY;BYDAY=SAT;BYHOUR=12',
start_date               => systimestamp at time zone 'US/Central',
comments                 => 'Saturday batch schedule',
schedule_name            => 'SAT BATCH'  ) ;
```

JOB CLASSES —The **RESOURCE_CONSUMER_GROUP** attribute associates a Resource Consumer Group with the Job Class. If not defined, it defaults to **DEFAULT_CONSUMER_GROUP**.

The **LOGGING_LEVEL** attribute for Job Classes can be specified as either:

- **LOGGING_OFF**: no logging
- **LOGGING_RUNS**: logs info on all runs in the class; default
- **LOGGING_FULL**: logs info on all runs plus operations like Enable, Disable, Alter, etc.

WINDOWS — Only one Window may be active at any given time. For overlapping Windows, the one with the higher priority remains active. For Windows with the same priority, the one that started first takes precedence.

This example code uses the **DBMS_SCHEDULER** procedure **CREATE_WINDOW** to create a Window. The **DURATION** is how long the Window is open for:

```
DBMS_SCHEDULER.CREATE_WINDOW(
window_name          => 'STD_PROCESSING',
resource_plan        => 'PROD_PLAN',
start_date           => systimestamp at time zone 'US/Central',
duration             => numtodsinterval(6, 'hour'),
repeat_interval      => 'FREQ=DAILY;BYHOUR=6',
window_priority      => 'HIGH',
comments             => 'evening batch time'  ) ;
```

PURGE_LOG procedure **—** This **DBMS_SCHEDULER** procedure purges the Scheduler logs. By default, once a day, all window and job logs are purged if they are more than 30 days old. The **LOG_HISTORY** attribute tells how many days of history to keep. The **WHICH_LOG** attribute tells which log to purge:

- **JOB_LOG**: for purging Job Logs only
- **WINDOW_LOG**: for purging Window Logs only
- **JOB_AND_WINDOW_LOG**: to purge both logs (this is the default)

This example code deletes both Job and Window log entries older than 14 days:

```
DBMS_SCHEDULER.PURGE_LOG(
log_history      => 14,
which_log        => 'JOB_AND_WINDOW_LOG' );
```

Key Privileges to use the new Scheduler:

Privilege:	**Meaning:**
CREATE JOB	For creating a Job, Program, or Schedule *in your own schema*
CREATE ANY JOB	For creating a Job, Program, or Schedule *in any user's schema*
EXECUTE ANY PROGRAM	Required to use components created by others
EXECUTE ANY CLASS	Gives user the ability to assign a Job to any Job Class
MANAGE_SCHEDULER system privilege	Required to create and manage Job Classes, Windows, and Window Groups
SCHEDULER_ADMIN Role	Includes CREATE JOB, CREATE ANY JOB, EXECUTE ANY PROGRAM, EXECUTE ANY CLASS, and MANAGE_SCHEDULER privileges. DBA Role has this Role by default

Windows and Window Groups have **PUBLIC** access, so everyone can access them without any special privileges.

Key Data Dictionary Tables:

Table:	Contains Information About:
DBA_SCHEDULER_JOBS	Scheduler Jobs
DBA_SCHEDULER_PROGRAMS	Scheduler Programs
DBA_SCHEDULER_JOB_CLASSES	Scheduler Job Classes
DBA_SCHEDULER_JOB_LOG	Log info on Scheduler Jobs
DBA_SCHEDULER_JOB_RUN_DETAILS	Log run details
DBA_SCHEDULER_RUNNING_JOBS	Scheduler Jobs that are currently running
DBA_SCHEDULER_JOB_ARGS	Scheduler Job's Arguments
DBA_SCHEDULER_SCHEDULES	Scheduler Schedules
DBA_SCHEDULER_WINDOWS	Scheduler Windows
DBA_SCHEDULER_WINDOW_GROUPS	Scheduler Window Groups
DBA_SCHEDULER_WINDOW_LOG	Scheduler Window Logs
DBA_SCHEDULER_WINGROUP_MEMBERS	Window Group Members

Many of these tables have **USER_** or **ALL_** equivalents. For example, for view **DBA_SCHEDULER_JOBS**, there are also views called **ALL_SCHEDULER_JOBS** and **USER_SCHEDULER_JOBS**.

New Data Movement Features

DBMS_FILE_TRANSFER Package

New package **DBMS_FILE_TRANSFER** contains 3 procedures that allow binary file copying locally or between a local and remote server. File size must be a multiple of 512 bytes and maximum file size is 2T.

The 3 new procedures are:

Procedure:	Use:
COPY_FILE	Copies a file on the local system
PUT_FILE	Copies a file on the local system to a remote system
GET_FILE	Copies a file on a remote system to the local system

PUT_FILE and **GET_FILE** require the use of a database link to refer to the remote system. For **PUT_FILE** this database link parameter is named **DESTINATION_DATABASE**, for **GET_FILE** this database link parameter is named **SOURCE_DATABASE**.

Check **V$DATAFILE** to see if the file was copied successfully.

Check **V$SESSION_LONGOPS** to monitor the copy in real-time.

External Table Enhancements_

External Tables are now writable (not just read-only as in 9i). You can populate them using parallelism, and also improve **SELECT** performance on them by using the Projected Columns feature. DML other than **SELECT** (**UPDATE**, **INSERT**, **DELETE**) and indexes are still not allowed on external tables.

9i only supported the **ORACLE_LOADER** access driver (or API), which can only read from external tables. 10g adds **ORACLE_DATAPUMP,** which allows writing to external tables. Only external tables created with **ORACLE_DATAPUMP** can be written to. These proprietary-format files can only be read subsequently by **ORACLE_DATAPUMP**. A key benefit of files created with **ORACLE_DATAPUMP** is that they can be used to load data into another Oracle database.

Use the **PARALLEL** keyword to specify the degree of parallelism you want for the external table. (This will typically be set equal to the number of files in the **LOCATION** parameter.)

Example of creating an external table with **ORACLE_DATAPUMP**:

```
    CREATE TABLE my_external_table
    ORGANIZATION EXTERNAL (TYPE ORACLE_DATAPUMP
    DEFAULT DIRECTORY my_directory
    LOCATION ('my_dump_1.dmp','my_dump_2.dmp') )
PARALLEL 2
AS
SELECT    -- statement here for the origin of the data --
```

If you know your external table data is clean, you can speed access to the data by making Oracle only process the columns referenced in the **SELECT** statement. Here's how to enable this feature:

```
ALTER TABLE my_external_table
PROJECT COLUMN REFERENCED ;
```

The default clause is **PROJECT COLUMN ALL**, which was the only option available in 9i. This feature accounts for the different number of total rows you may retrieve from a table with a **SELECT** statement (depending on your **PROJECT COLUMN** setting and whether you refer to any columns containing bad data in your queries).

Transportable Tablespaces Across Platforms

You can now use transportable tablespaces to move data across platforms. However, different platforms may have different endian formats – some store the most significant bytes first while others store them last. So you may have to run RMAN's **CONVERT TABLESPACE** command to convert the different endian formats between platforms.

Here is the procedure for cross-platform transportable tablespaces:

1. First find out the endian formats for the platforms involved by querying view **V$TRANSPORTABLE_PLATFORM**:

   ```
   SELECT platform_id, platform_name, endian_format
   FROM V$TRANSPORTABLE_PLATFORM
   ORDER BY platform_name ;
   ```

   V$DATABASE also contains new columns **PLATFORM_ID** and **PLATFORM_NAME**.
2. Check that the tablespaces are self-contained by running procedure **DBMS_TTS.TRANSPORT_SET_CHECK**.
3. Alter the tablespaces to be transported to read-only in the Source database:

   ```
   ALTER TABLESPACE transport_me READ ONLY;
   ```
4. Use the **expdp** utility to unload the metadata for the transportable tablespace(s) by specifying the TRANSPORTABLE_TABLESPACE parameter.
5. If endian formats were found to be different in Step (1), convert those datafiles by using RMAN's **CONVERT TABLESPACE** command. The **TO PLATFORM** clause specifies the Target platform name, which must be spelled ***exactly*** as it appears in **V$TRANSPORTABLE_PLATFORM**.
6. Copy the RMAN-converted datafiles and metadata dump files to the Target system.
7. Use the **impdb** utility to import the metadata and plug in the tablespaces.
8. Alter the new tablespaces to be read / write in the Target database:

   ```
   ALTER TABLESPACE transport_me READ WRITE;
   ```

Data Pump

Data Pump (or DataPump) supersedes the Export and Import utilities of 9i (which are still present in 10g). Data Pump's Export and Import utilities are called **expdp** and **impdp**, respectively.
Advantages to Data Pump:

- Parallelism (example: specify **PARALLEL = 2**)
- Much faster than traditional Export/Import
- Supports job restart and monitoring. User sessions can detach and **ATTACH** again to jobs in real-time
- Supports two access methods:
  - Direct Path access
  - External Table access
- Data Pump selects the optimal access method automatically for you – you cannot dictate this choice! Data Pump selects the Direct Path API whenever possible. The output file format is the same for both access methods so their use can be intermingled. Data Pump cannot load data into a database that was exported with the older pre-10g Export utility (exp)
- Data Pump runs on the Server (not a Client)
- Uses two key packages: **DBMS_METADATA** and **DBMS_DATAPUMP**

Data Pump Components:

Component:	**Use:**
Master Control Process (MCP)	Directs the Data Pump job and manages the other processes. There is one MCP per Data Pump job.
Master Table	Created and maintained by the MCP in the schema of the user who runs the job. Data Pump writes it to the end of the dump file after an export and reads it into the schema of the user at the start of an import. The Master Table has the same name as that of the Data Pump job.
Worker Processes	MCP creates a number of these (based on the **PARALLEL** parameter value) to load/unload data.
Parallel Query (PQ) Processes	Created by Worker Processes as needed for external table access.
Client Process	Started by the client utility **expdp** or **impdp.**
Advanced Queuing (AQ)	Data Pump uses AQ facilities and queues for internal communications.

Data Pump may write 3 types of files:

- Dump files: contains data and metadata
- Log files: tracks job activities
- SQL files: contains SQL statements extracted

You can specify where these files are written to in 3 ways:

- Use the **DIRECTORY** parameter in the **expdp** or **impdp** utilities
- Encode it in the **DUMPFILE** parameter like this: **DUMPFILE=my_dump_directory:my_file.dmp**
- Define **DATA_DUMP_DIR** as an OS environment variable that points to a defined directory in the database

You cannot write to absolute or fully qualified path names, only to directories as described by the 3 ways above. There are many parameters to the **expdp** and **impdp** utilities. These lists focus on the key parameters for the test.

EXPDP Parameters:

Parameter:	**Use:**
ATTACH	Connects or re-connects a client to a Data Pump session or job
CONTENT	**ALL** (default), **DATA_ONLY** or **METADATA_ONLY**. ***BOTH** is **not** an allowable parameter!*
DIRECTORY	Names a database directory object for output
DUMPFILE	File to store export data in. Use **%U** to specify more than one file with the same name, **expdp** expands this to a unique number. Cannot be specified with the **ESTIMATE_ONLY** parameter
ESTIMATE	Specifies either **BLOCKS** or **STATISTICS** as the way to estimate export disk space. The Default is BLOCKS
ESTIMATE_ONLY	Estimates the disk space needed for export -- **but does *not* perform the export. Can *not* be specified with the DUMPFILE parameter**
EXCLUDE	Lists metadata objects excluded from export. This helps you to specify the objects for export
FILESIZE	Maximum file size for each dump file
FLASHBACK_SCN	Export the data as of this consistent System Change Number (SCN)
FLASHBACK_TIME	Export the data as of this consistent Time
FULL	= Y means export the full Database

HELP	Display Help info
INCLUDE	Lists metadata objects included in the export. This helps you specify the objects for export
JOB_NAME	The Job name
KEEP_MASTER	= Y means keep the Master Table after the job completes
LOGFILE	Name of the log file
NETWORK_LINK	Use this Database Link to perform the export. *This allows you to export directly to another database server*
PARALLEL	Maximum number of parallel threads for export
PARFILE	Parameter file (may not be nested)
QUERY	Adds a **WHERE** clause for export selectivity. Far more advanced options than the old exp query parameter
SCHEMAS	Names of the users/schemas to export
TABLES	Lists Tables to export
TABLESPACES	Lists the Tablespaces to export
TRANSPORT_FULL_CHECK	= Y means check dependencies for transportable tablespaces
TRANSPORT_TABLESPACES	= Y for transportable tablespace mode
VERSION	**Only** exports objects compatible with the specified database version

Many of the above parameters are also common to the Import (**impdp**) utility. The table below shows some other parameters for **impdp**.

IMPDP Parameters:

Parameter:	Use:
FLASHBACK_SCN	Import data consistent with this Systems Change Number. **Valid only when NETWORK_LINK is specified**
FLASHBACK_TIME	Import data consistent with the Time specified. **Valid only when NETWORK_LINK is specified**
NETWORK_LINK	Imports from the source database using this Database Link
REUSE_DATAFILES	Overwrites existing datafiles
SKIP_UNUSABLE_INDEXES	= Y means skip loading tables with bad indexes and continue the job
SQLFILE	Metadata SQL statements are written to this file
STATUS	Operational status
TABLE_EXISTS_ACTION	Options are expanded in 10g and now include: **APPEND, REPLACE, SKIP, TRUNCATE**
TRANSPORT_DATAFILES	Datafiles to import via transportable tablespaces method
REMAP_DATAFILE	Changes source Datafile name
REMAP_SCHEMA	Changes source Schema name
REMAP_TABLESPACE	Changes source Tablespace name
TRANSFORM	Changes object creation attributes. *Relies on a boolean value for the transform*

The final 4 items above allow for object transformation:

- **REMAP_DATAFILE**
- **REMAP_SCHEMA**
- **REMAP_TABLESPACE**
- **TRANSFORM**

(*There is **no** such thing as REMAP_USER !)*

These transformations work because exported metadata is stored as XML.

Do not confuse any of the older **exp/imp** parameters with the new **expdp/impdp** parameters! Some old **exp/imp** parameters that are not valid for **expdp/impdp** include: OWNER, FEEDBACK, FILE, TTS_FULL_CHECK, FROMUSER, TOUSER, DESTROY, DATAFILES, and IGNORE.

Exports and Imports can be performed on 5 levels:

- Database: **FULL = Y**
- Tablespace: **TABLESPACES** = list
- Schema: **SCHEMAS** = list
- Table: **TABLES** = list
- Transport Tablespace: **TRANSPORT_TABLESPACES** = list

Interactive commands for managing Data Pump jobs:

Command:	**Use:**
ADD_FILE	Adds a file to the **DUMPFILE** set
CONTINUE_CLIENT	Changes from interactive client to logging mode
EXIT_CLIENT	Exits client session but leaves the job running
KILL_JOB	Stops the job, **it cannot be started later**
PARALLEL	Increases or decreases number of parallel threads
START_JOB	Starts or re-starts the job
STOP_JOB	Stops the job, **it can be re-started later**
STATUS	Displays job status

Data Pump dictionary views include:

- **V$SESSION_LONGOPS**: preferred view for info on job progress
- **DBA_DATAPUMP_JOBS**: active Data Pump job information
- **DBA_DATAPUMP_SESSIONS**: session information
- **DATAPUMP_PATHS**: valid object types for Export/Import

Automated Management

The single most defining characteristic of 10g is that it manages itself through automated means. This is called the Common Manageability Infrastructure (CMI), and it encompasses system-wide statistics collection, memory management, storage management, self-diagnostics, and many other features.

A key part of CMI is the Advisory Framework, a set of components that provides advice on how to tune Oracle. The Automatic Database Diagnostic Monitor (or ADDM) is a built-in, real-time, self-diagnostic performance monitor. Other components in the Advisory Framework automatically tune memory: the PGA Advisor, the Buffer Cache Advisor, and the Shared Pool Advisor. The Segment Advisor and the Undo Advisor help tune storage. And the SQL Tuning Advisor and the SQL Access Advisor help you tune data access through SQL statement and database design advice, respectively.

Figure 2: Common Manageability Infrastructure

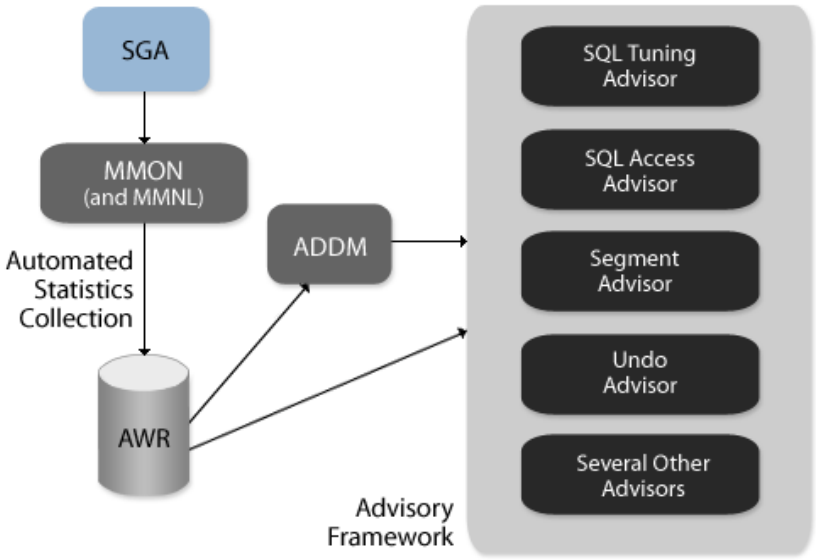

Figure 2 diagrams the CMI and its relationship to the Advisory Framework. The CMI will be discussed in this section. Later sections go into detail on the remaining Advisors.

Performance Statistics

10g supersedes 9i's Statspack for database statistics collection and analysis with Automatic Optimizer Statistics Collection. 10g's new approach is not compatible with Statspack and provides no migration of or integration with Statspack statistics.

10g statistics are collected into a memory buffer that is allocated within the Systems Global Area (or SGA), the basic memory area used by Oracle. Statistics are then written from the SGA buffer and stored onto a disk in the new Automated Workload Repository (or AWR). The AWR physically resides in the new required **SYSAUX** tablespace. Data is owned by **SYS** and may be viewed through new data dictionary and V$ views.

The new background process **MMON** (Manageability Monitor) writes statistics from the SGA buffer to the AWR by default once every hour. These are hourly snapshots. The new background process **MMNL** (Manageability Monitor Light) flushes the statistics to the AWR whenever the SGA buffer area that holds them becomes full.

Thus, snapshots of statistics are written to AWR by MMON at hourly intervals or by MMNL whenever the SGA memory buffer collection area becomes full. The process is fast because it is fully integrated into Oracle internals. AWR performance statistics and workload information is saved by default for **7 days**. The new **STATISTICS_LEVEL** initialization parameter determines the level of statistics collection and analysis and may be set to any of 3 values:

- **BASIC**: disable AWR statistics and self-tuning capabilities
- **TYPICAL**: default level of statistics collection
- **ALL**: collect exhaustive statistics

To turn off AWR statistics and self-diagnostics, set **STATISTICS_LEVEL = BASIC**.
The statistics AWR collects include:

- **Time model statistics** that indicate time spent in various activities
- **Object statistics** on database feature usage
- **V$SYSSTAT** and **V$SESSTAT** wait class statistics
- **High-load SQL** statements
- **Operating System** statistics

In addition to AWR's hourly snapshots of the above information, Active Session History (or ASH) samples the sessions activity in view **V$SESSION** every second. ASH fills the vacuum of activity left by AWR, which operates on an hourly basis, since ASH samples **V$SESSION** every second.

Query view **V$ACTIVE_SESSION_HISTORY** for ASH information, it contains one row for each active session per sample. Or query the higher-level view **DBA_HIST_ACTIVE_SESS_HISTORY**, which contains a sampled history of **V$ACTIVE_SESSION_HISTORY** information.

Do not confuse these AWR or ASH statistics with traditional optimizer statistics. Optimizer statistics are the statistics Oracle collects on tables and indexes in order to determine optimal data access strategies or access paths. Optimizer statistics are a topic that will be covered later in the Exam Manual.

Query view **DBA_HIST_DATABASE_INSTANCE** to see which database instances the AWR tracks. If you have the **SELECT ANY DICTIONARY** privilege, you can run any of the 3 key AWR reports:

- **awrrpt.sql**: reports detailed statistics for a range of snapshot IDs in either text or HTML formats
- **awrrpti.sql**: reports the same but you specify the database and instance IDs as input
- **awrinfo.sql**: general information on AWR snapshots and ASH use

The **DBMS_WORKLOAD_REPOSITORY** package allows you to manage snapshots yourself. You can also manage baselines, which are measurement points derived from a set of snapshots. Baselines are used for comparison purposes when a problem occurs.

DBMS_WORKLOAD_REPOSITORY procedures include:

- **CREATE_SNAPSHOT**: manually creates a snapshot at some time other than the standard 1-hour interval.
- **DROP_SNAPSHOT_RANGE**: drops a range of snapshots by specifying low and high snapshot IDs. Also drops the ASH history within that time frame from **DBA_HIST_ACTIVE_SESS_HISTORY**.
- **CREATE_BASELINE**: creates a baseline based on a range of snapshot IDs.
- **DROP_BASELINE**: drops a baseline by name. The **CASCADE** parameter drops snapshots related to the baseline you drop (the default for **CASCADE** is **FALSE**, so baseline-related snapshots are not dropped by default). The snapshots belonging to a baseline are not dropped until the baseline is dropped.
- **MODIFY_SNAPSHOT_SETTINGS**: modifies the snapshot capture and purging policy. Two key parameters to remember:
  - **RETENTION** - the new retention period in minutes
  - **INTERVAL** - the new snapshot interval in minutes

Here are a couple examples of the DBMS_WORKLOAD_REPOSITORY procedures. Using **MODIFY_SNAPSHOT_SETTINGS,** this example sets the **RETENTION** period for snapshots to 2,000 minutes:

- DBMS_WORKLOAD_REPOSITORY**.**MODIFY_SNAPSHOT_SETTINGS- (RETENTION=>2000);

This example sets the snapshot **INTERVAL** to 20 minutes:

- DBMS_WORKLOAD_REPOSITORY**.**MODIFY_SNAPSHOT_SETTINGS- (INTERVAL=>20);

This statement creates a snapshot on demand:

- DBMS_WORKLOAD_REPOSITORY**.**CREATE_SNAPSHOT() ;

This statement drops the baseline named 'my_base':

- DBMS_WORKLOAD_REPOSITORY**.**DROP_BASELINE -
  (BASELINE_NAME=>'my_base');

The key dictionary views for snapshot information are:

- **DBA_HIST_SNAPSHOT**: snapshot info with startup time and details
- **DBA_HIST_BASELINE**: lists snapshot IDs and baseline time range
- **DBA_HIST_WR_CONTROL**: lists current snapshot settings for **RETENTION** and **INTERVAL**

Base statistics are raw data. Metrics describe base statistics in terms of time frames. Background process **MMON** creates and updates metric data.

Server-based Alerts

The 10g CMI includes server-generated alerts. These trigger either because:

1. A threshold (or value) is exceeded
2. An event has occurred

MMON sends Alerts to the predefined Alert Queue owned by **SYS** named **ALERT_QUE**. They also display on the Database Control homepage and get sent to administrators via **email** or **pager** (depending on your EM configuration).

Threshold alerts are also known as stateful alerts. They must occur a specified number of times within a time period to be raised (this avoids false alerts). Threshold alerts appear in view **DBA_OUTSTANDING_ALERTS** and when cleared are moved to view **DBA_ALERT_HISTORY** with a status of **CLEARED**.

Non-threshold alerts are better known as event alerts or stateless alerts. These go directly to **DBA_ALERT_HISTORY** and never appear in **DBA_OUTSTANDING_ALERTS.**

Oracle provides 4 "out-of-the-box" alerts:

- Recovery area low on free space
- Snapshot too old
- Resumable session suspended
- Tablespace space is running out
    - **85% used** means a **Warning message**
    - **97% used** means a **Critical message**

Important views are:

- **V$ALERT_TYPE**: lists all the possible alert types and whether each is threshold or stateless
- **DBA_THRESHOLDS**: shows the threshold settings

You would normally use the EM GUI to view and change alerts. However, the test concentrates on coding, so remember the package **DBMS_SERVER_ALERT** and its two procedures:

- **GET_THRESHOLD**: retrieve current threshold value
- **SET_THRESHOLD**: define a new threshold value

Automatic Database Diagnostic Monitor (ADDM)

MMON triggers ADDM analysis each time a snapshot is taken on the last two snapshots. This proactively monitors the database for any developing problems. Analysis is top-down with the goal of reducing everything to the single metric **DB time** (time spent on processing user requests).

Findings divide into *problem* (root cause), *symptom*, or *information*. ADDM may recommend running one of the other Advisors, such as the SQL Tuning Advisor.

Access ADDM findings through EM (off the Performance link) or through view **DBA_ADVISOR_FINDINGS**. You can also run report scripts **addmrpt.sql** or **addmrpti.sql** (takes Database and Instance IDs as input parameters).

DBA_ADVISOR_RECOMMENDATIONS shows the result of diagnostics runs with recommendations for the problems identified in each run.

You can set a few of the ADDM parameters by using **DBMS_ADVISOR.SET_DEFAULT_TASK_PARAMETER**. View current parameters through view **DBA_ADVISOR_DEF_PARAMETERS**.

Like all 10g's automation features, ADDM is enabled by default (because by default **STATISTICS_LEVEL** is **TYPICAL**). A **STATISTICS_LEVEL** setting of **ALL** also keeps ADDM on while the setting of **BASIC** turns it off.

Automatic Shared Memory Management (ASMM)

SGA_TARGET is a new dynamic initialization parameter that specifies the total size of SGA memory. Setting this to a non-zero value (plus setting **STATISTICS_LEVEL** to **TYPICAL** or **ALL**) enables ASMM. ASMM then takes responsibility for dynamically sizing several parts of the SGA as appropriate. **SGA_TARGET** cannot be larger than non-dynamic initialization parameter **SGA_MAX_SIZE**.

Memory components for which ASMM automatically configures memory are:

- **DB_CACHE_SIZE**: database buffer cache default pool
- **SHARED_POOL_SIZE**: Shared pool
- **LARGE_POOL_SIZE**: the Large pool
- **JAVA_POOL_SIZE**: the Java pool

ASMM does **not** size these areas, which you should manually size through initialization parameters:

- **LOG_BUFFER**: the log buffer
- **STREAMS_POOL_SIZE**: size of the Streams pool
- Fixed-size SGA components and internal allocations
- Other buffer caches:
    - **DB_nK_CACHE_SIZE**
    - **DB_KEEP_CACHE_SIZE**
    - **DB_RECYCLE_CACHE_SIZE**

Here's an example of how this works — You set **SGA_TARGET** to 800 M, set **LOG_BUFFER** to 10 M, and set **DB_KEEP_CACHE_SIZE** to 100 M. This means ASMM distributes 800 - (10 + 100) or 690 M to the memory areas it dynamically sizes.

When **SGA_TARGET** is set to a non-zero value, the auto-tuned SGA parameters will have default values of 0. If you specify a non-zero value for any auto-tuned parameter, ASM takes that value as the lower limit for that component. Remember that setting **SGA_TARGET** to 0 disables ASMM. The new background process Memory Manager (MMAN) does the memory sizing for ASMM.

You can see ASMM and memory configuration information on the Memory Parameters page of the EM GUI, or see view **V$SGA_DYNAMIC_COMPONENTS**.

Mean Time to Recovery (MTTR) Advisor and Automatic Checkpoint

Recall that initialization parameter **FAST_START_MTTR_TARGET** indicates the time, in seconds, that instance recovery should take after a crash or instance failure.

To enable automatic checkpointing, set the **FAST_START_MTTR_TARGET** initialization parameter to a non-zero value. Disable it by setting **FAST_START_MTTR_TARGET** to 0. (As always, **STATISTICS_LEVEL** must be either **TYPICAL** or **ALL** to enable any of the Advisors).

When enabled, this feature tries to write dirty buffers (updated database pages or blocks) without impacting performance and to ensure reasonable crash-recovery time.

View **V$MTTR_TARGET_ADVICE** shows expected I/O for different **FAST_START_MTTR_TARGET** values after the workload has been executing for awhile.

Note that you can still set parameters **FAST_START_IO_TARGET** and **LOG_CHECKPOINT_INTERVAL** to control instance recovery — but setting either one will also disable **FAST_START_MTTR_TARGET**.

Redo Logfile Size Advisor

Online Redo Logs should be sized large enough so that checkpoints do not occur too frequently (more than a log switch every 20 minutes), but small enough so that you do not use up disk space without any noticeable performance benefit.

The Redo Logfile Size Advisor helps with this.

View **V$INSTANCE_RECOVERY** has a new column **OPTIMAL_LOGFILE_SIZE** that shows the optimal size of the Redo Log files. The online redo logs must be set to at least the size recommended. The EM GUI also provides this same information.

For the Redo Logfile Size Advisor to provide optimal Logfile sizing, the MTTR Advisor must also be enabled and active.

Automatic Undo Retention

The new Undo Advisor analyzes undo use and recommends the undo tablespace size needed to support the longest-running query when using automatic undo.

This obsoletes these parameters from earlier Oracle releases:

- MAX_ROLLBACK_SEGMENTS
- UNDO_SUPPRESS_ERRORS
- ROW_LOCKING
- SERIALIZABLE
- TRANSACTION_AUDITING

If you set the initialization parameter **UNDO_RETENTION** to 0 or if no value is specified, 10g automatically tunes the undo retention for the undo tablespace using 900 seconds as the minimum value. Setting **UNDO_RETENTION** to a value other than 0 causes 10g to use that number of seconds as the minimum value.

Setting a value for the new keyword **RETENTION_GUARANTEE** causes Oracle to guarantee that undo will always be available for the specified retention period. Specify **RETENTION_GUARANTEE** clause when creating the Undo Tablespace. View the current retention value in the view **DBA_TABLESPACES**. Or use EM. Be sure you know how to read the EM GUI output for optimally sizing the undo.

Other Advisors

The Program Global Area (PGA) Advisor auto-tunes PGA memory and recommends a value for initialization parameter **PGA_AGGREGATE_TARGET**.

The Segment Advisor analyzes disk space and provides recommendations and a Growth Trend Analysis. The SQL Access Advisor provides tuning recommendations related to database design and data access issues while the SQL Tuning Advisor tunes SQL statements.

These Advisors will be discussed in detail in the appropriate sections below.

Managing the Advisors

Initialization parameter **STATISTICS_LEVEL** must be **TYPICAL** (the default) or **ALL** to enable the Advisors. A setting of **BASIC** disables them.

Manage all the Advisors through the EM GUI, or by the **DBMS_ADVISOR** package. You need the **ADVISOR** privilege to manage the Advisors.

The typical procedure working with the Advisors from the command line is:

1. Use **DBMS_ADVISOR.CREATE_TASK** to create an advisor task
2. Use **DBMS_ADVISOR.SET_TASK_PARAMETERS** to set parameters that control the advisor's behavior
3. Run analysis by **DBMS_ADVISOR.EXECUTE_TASK**
4. View analysis by **DBMS_ADVISOR.GET_TASK_REPORT**

Key Advisor dictionary views:

DBA_ADVISOR_ACTIONS	Info on the actions related to the recommendations
DBA_ADVISOR_COMMANDS	Commands used by all Advisors
DBA_ADVISOR_DEFINITIONS	Lists names of the Advisors
DBA_ADVISOR_FINDINGS	Advisor findings
DBA_ADVISOR_JOURNAL	Journal entries for all tasks
DBA_ADVISOR_LOG	Log info with current state of all tasks
DBA_ADVISOR_OBJECTS	Objects the Advisors refer to
DBA_ADVISOR_PARAMETERS	Parameter values for all tasks
DBA_ADVISOR_RATIONALE	Rationales behind the recommendations. Look here to understand how an Advisor came up with its recommendations
DBA_ADVISOR_RECOMMENDATIONS	The recommendations that come out of an analysis

Simplified Shared-Server Configuration

Setting the new initialization parameter **SHARED_SERVERS** to any value greater than 0 enables a shared-server configuration. This is the default for 10g; one TCP dispatcher starts automatically.

Shared-server configuration manages multiple user processes/requests through each shared server process.

The many initialization parameters that start with the letters **MTS_** are now obsolete (e.g.: **MTS_SERVERS**, **MTS_DISPATCHERS**, etc.).

Here are the obsolete parameters and their 10g equivalents:

Obsolete Initialization Parameter:	**New 10g Equivalent:**
MTS_SERVERS	SHARED_SERVERS
MTS_MAX_SERVERS	MAX_SHARED_SERVERS
MTS_DISPATCHERS	DISPATCHERS
MTS_MAX_DISPATCHERS	MAX_DISPATCHERS
MTS_SERVICE	SERVICE_NAMES

MTS_SESSIONS	SHARED_SERVER_SESSIONS
MTS_LISTENER_ADDRESS	LOCAL_LISTENER
MTS_MULTIPLE_LISTENERS	LOCAL_LISTENER
MTS_CIRCUITS	CIRCUITS

The new view **V$DISPATCHER_CONFIG** provides information on dispatchers.

Automated SQL Management

Optimizer Statistics

Optimizer statistics are those statistics on tables and indexes that Oracle requires in order to create the most efficient data access paths. 10g fully automates the collection of optimizer statistics for the first time. Remember that the default **STATISTICS_LEVEL** enables statistics with its default value of **TYPICAL** and that table monitoring is enabled by default.

Creating a new 10g database through DBCA, or updating an existing database to 10g through DBUA, automatically creates a job by the name of **GATHER_STATS_JOB**. This job executes the procedure: **DBMS_STATS.GATHER_DATABASE_STATS_JOB_PROC**, which gathers statistics on an object when its statistics are stale because more than 10% of its rows have been modified. This procedure is intelligent in that it updates the most needed statistics first. It is similar to the procedure **DBMS_STATS.GATHER_DATABASE_STATS** with the **GATHER_AUTO** option in previous Oracle releases.

The **MAINTENANCE_WINDOW_GROUP** is automatically created, which contains two Windows:

- **WEEKEND_WINDOW**: Starts Mon thru Fri at 2200 hours, goes for 8 hours
- **WEEKNIGHT_WINDOW**: Starts Sat at 0000 hours, goes for 48 hours

If you create a new database — but not through DBCA or DBUA — then the statistics job is still created but it is **not** scheduled to run.

In previous releases, you did not gather statistics on (1) dictionary objects and (2) fixed tables (Fixed tables are Oracle tables that exist only in memory and are created, used, and managed by Oracle). Either the **SYSDBA** or new system privilege **ANALYZE ANY DICTIONARY** is required to gather statistics for the dictionary and fixed tables.

In 10g, you can gather dictionary table statistics by any of these 3 **DBMS_STATS** procedures by using the option **GATHER_AUTO**:

- **GATHER_DATABASE_STATS**
- **GATHER_SCHEMA_STATS**
- **GATHER_DICTIONARY_STATS**

You would gather statistics only once for the fixed tables, during a normal workload that has been running for a few minutes. To get fixed table statistics, run procedure **DBMS_STATS.GATHER_DATABASE_STATS** with the parameter **GATHER_FIXED** set to **TRUE** (its default is **FALSE**). Or run **DBMS_STATS.GATHER_FIXED_OBJECTS_STATS** to collect statistics on the fixed tables only.

10g allows you to lock statistics for a particular table or schema. Statistics are not collected or updated for locked tables and schemas during statistics collection. The view **DBA_TAB_STATISTICS** has the column **STATTYPE_LOCKED**, which will be set to **ALL** to indicated locked statistics.

Use procedures **DBMS_STATS.LOCK_TABLE_STATS** and **DBMS_STATS.LOCK_SCHEMA_STATS** to lock statistics. Use procedures **DBMS_STATS.UNLOCK_TABLE_STATS** and **DBMS_STATS.UNLOCK_SCHEMA_STATS** to unlock them and resume statistics collection on those entities.

You can use the **FORCE => TRUE** parameter to override statistics locking in several **DBMS_STATS** procedures. In other words, **FORCE => TRUE** collects statistics even for locked objects.

Old statistics are automatically saved with the AWR, even as new ones are collected. They can be manually restored if you ever need them. View **DBA_TAB_STATS_HISTORY** shows how the statistics for tables have been updated.

Old statistics are automatically purged from the AWR at the default interval of 31 days. Note that if you use the old pre-10g **ANALYZE** statement, old versions of statistics are not saved in the AWR and are not available for restore.

Here are some examples of collecting statistics. To collect data dictionary statistics, issue:

- DBMS_STATS.GATHER_DICTIONARY_STATS ;
  or
- DBMS_STATS.GATHER_SCHEMA_STATS('SYS') ;

To gather statistics on the fixed tables, you might issue:

- DBMS_STATS.GATHER_FIXED_OBJECTS_STATS ;
  or
- DBMS_STATS.GATHER_SCHEMA_STATS('SYS',GATHER_FIXED=>TRUE)

The Rule-based Optimizer is de-supported, so you can now only use the Cost-based Optimizer. Initialization parameter **OPTIMIZER_MODE** therefore no longer supports the options **RULE** and **CHOOSE**. The allowable values are now:

- **FIRST_ROWS**: deliver the first few rows quickly
- **FIRST_ROWS_n**: **n** must be 1, 10, 100, or 1000
- **ALL_ROWS**: the default, optimize best for returning the entire results set

The 10g cost-based optimizer (or CBO) considers both CPU and I/O in making its decisions. The **PLAN_TABLE**'s new column labeled **TIME** estimates the elapsed time for any query, in seconds.

The new initialization parameter **OPTIMIZER_DYNAMIC_SAMPLING** can be set between 0 and 10 (inclusive) in order to determine how much dynamic sampling occurs when an execution plan is generated. This can be useful for objects that have no statistics or that have out-dated statistics. The default value is 2. Set it higher for more sampling in appropriate situations.

SQL Tuning Advisor

The new SQL Tuning Advisor automates SQL statement tuning. It operates in two modes:

- Normal
- Tuning

The former is real-time, while the latter takes minutes for its analysis rather than seconds. Tuning mode provides recommended actions (and their rationales) to produce better SQL execution. The tuning mode is also called the Automatic Tuning Optimizer (ATO). This mode is great for improving high-load SQL. The SQL Tuning Advisor has these main inputs:

- High-load SQL (identified by ADDM)
- SQL statements in the cursor cache
- SQL statements from the AWR
- A user-provided set of SQL statements
- A SQL Tuning Set (or STS)

User-defined SQL statements are not executed; they are merely analyzed. STS is a set of SQL statements along with execution information (bind variables, how much to execute the SQL, etc.).

The kinds of analysis the SQL Tuning Advisor does is:

- Statistics analysis: ATO verifies statistics, recommends gathering new ones if appropriate
- SQL profiling: a SQL Profile includes info collected during optimization useful for further optimization during future runs; enables tuning of queries without requiring any syntax changes
- Access path analysis: on a single-statement level, may recommend indexes
- SQL structure analysis: inspects the keywords used and the structure of the SQL statement; recommends improvements

You can work with the SQL Tuning Advisor through EM Database Control or through the package **DBMS_SQLTUNE.** To use this package, a typical set of steps would be:

1. Use **DBMS_SQLTUNE.CREATE_TUNING_TASK** to create a task
2. Run **EXECUTE_TUNING_TASK**
3. Accept the SQL profile generated by **ACCEPT_SQL_PROFILE**

You need the **ADVISOR** privilege to do this, as well as the **CREATE ANY SQL PROFILE** privilege to create and save SQL Profiles. SQL Profiles are uniquely identified by a category name and SQL text, so you can have identical SQL statements in different profiles as long as each profile is in a different category. Categories are determined by the new initialization parameter **SQLTUNE_CATEGORY**. The default category is called **DEFAULT**. You can change the category at the session level with a statement like:

```
ALTER SESSION SET SQLTUNE_CATEGORY = PROD ;
```

Another useful **DBMS_SQLTUNE** procedure is **QUICK_TUNE**. Use it to tune a single SQL statement. Here is an example of how to use it:

```
VARIABLE        task_id    NUMBER ;
VARIABLE        task_name          VARCHAR2(255) ;
VARIABLE        sql_stmt  VARCHAR2(4000);
EXECUTE         :sql_stmt  :=  'SELECT COUNT(*) FROM emp WHERE empno=1434';
EXECUTE         :task_name         := 'QUICKTUNE';
EXECUTE DBMS_ADVISOR.QUICK_TUNE('SQL Access Advisor', -
                :task_id, :task_name, :sql_stmt ) ;
```

SQL Access Advisor

The SQL Access Advisor helps determine which indexes, materialized views, and materialized view logs will aid one query or a workload of queries. It differs from the SQL Tuning Advisor in that its suggestions are more directed to database design issues, rather than tuning the SQL statement's structure and optimization.

SQL Access Advisor operates in two modes:

- Limited
- Comprehensive

The latter analyzes a full or complete workload.

SQL Access Advisor input can come from:

- SQL statements from the SQL cache (**V$SQL**)
- A user-provided set of SQL statements
- A SQL Tuning Set (or STS)
- A schema

Like SQL Tuning Advisor, you can access the SQL Access Advisor through the Performance Tab of EM Database Control. Other performance information you get through the EM Database Control panel include three areas:

1. CPU use
2. Top SQL statements
3. Top sessions

This GUI gives you a quick instance-wide overview, allowing you to then invoke Advisors as appropriate to better understand and resolve any problems. Look for long wait events as bottlenecks; Oracle should always be busy, not waiting on events.

Automatic Storage Management (ASM)

In 10g, Automatic Storage Management, or ASM, offers an entirely new way to manage Oracle storage. ASM is highly automated; it provides features like striping and mirroring; it supports all kinds of filesystems, and it often eliminates the need to purchase a third-party volume management tool or clustered filesystem support tool. ASM manages files and provides an alternative to raw devices or filesystems locally attached to a database instance. It supports Real Application Clusters (RAC) if desired.

ASM uses an entire new instance, the ASM instance, to control datafiles and storage. Other database instances called "clients" use the ASM instance for their storage needs. If you use OUI and the DBCA to create a database, it provides an option to automatically create the ASM instance for you.

The ASM instance does not have a data dictionary. All connections into it must be via operating system authentication. So you connect into it with the **SYSDBA** or **SYSOPER** login ids defined by an operating system user in the **dba** group. The ASM instance does not have a control file, so you do not supply an initialization parameter that specifies a control file to an ASM instance.

You start the ASM instance first, then the client instances next. You shutdown the ASM instance first, which then propagates the shutdown to its client instances. For example, issue a **SHUTDOWN NORMAL** to the ASM instance, and it propagates this command to its clients. The ASM instance waits for its clients to shutdown before completing.

The exception to this behavior is when you issue **SHUTDOWN ABORT** to the ASM instance. Issuing this command to the ASM instance causes ASM to abort without passing the command to its clients. (But they'll abort anyway, because now their storage is missing.)

You can issue **STARTUP RESTRICT** to the ASM instance to start it without allowing any client instances to connect to it.

ASM Storage

ASM uses the concept of a disk group, a set of disk devices it manages as a logical unit. (Disk groups are analogous to the "volume groups" used by many operating systems to group and manage disks.) ASM divides storage into a most basic unit called extents. Extents are the unit of data mirroring.

ASM automatically rebalances data across a disk group when space is added to or removed from a disk group. The ASM background process **RBAL** coordinates the rebalancing, while the processes **ARBn** (**n** is from 0 to 9) actually carry out the rebalancing. Rebalancing occurs transparent to users in the background. The ASM instance initialization parameter **ASM_POWER_LIMIT** dictates how fast rebalancing occurs and how much resource it consumes. Set **ASM_POWER_LIMIT** between 1 and 11 inclusive, where the default value is 1 (lowest overhead) and 11 is the highest permissible value. This parameter is dynamic, so you could, for example, stop an in-progress rebalancing operation by setting it to 0 in real-time.

Data striping across disk groups is done in either of two modes:

- **Coarse**: units of 1 M each; cannot be used for controlfiles, redo logs, or flashback files
- **Fine**: units of 128 K each

Mirroring is performed by the basic unit of the extent. There are 3 kinds of mirroring allowed:

Mirroring Type:	Mirroring:	Failure Groups Required:
High Redundancy	3 -way mirroring	3
Normal Redundancy	2 -way mirroring	2
External Redundancy	n/a	1

A failure group is a set of disks within a disk group that would fail as a unit. An example is – a set of disks on the same controller. So, High Redundancy means that 2 of the 3 failure groups could fail and the data would still be accessible. Normal Redundancy only allows for failure of 1 failure group, since it is based on a two-way mirror. External Redundancy requires only a single failure group. Either the resource is not critical enough for mirroring, or you are relying on mirroring or facilities external to Oracle (for example, in the operating system or an enterprise storage subsystem).

ASM offers several commands to manage disk groups (which can also be performed from the EM GUI). Contents of the views vary slightly between the ASM and database client instances.

This information is described as selected from within the ASM instance:

View:	Use:
V$ASM_CLIENT	Lists the database clients using ASM's disk groups
V$ASM_DISK	Lists all ASM disks (whether in a disk group or not)
V$ASM_DISKGROUP	Lists the disk groups
V$ASM_FILE	Lists the files in every mounted disk group
V$ASM_OPERATION	One row for each long-running ASM operation. Look here to see the status of ASM operations
V$ASM_TEMPLATE	Lists the templates for the disk groups
V$ASM_ALIAS	Lists aliases for mounted disk groups

So to manage disks and diskgroups, query **V$ASM_DISK** to see what disks are available to ASM, and which are presently in a disk group. Use the **CREATE DISKGROUP** statement to create a disk group:

```
CREATE DISKGROUP my_group1 HIGH REDUNDANCY
FAILGROUP fg1     DISK '/dev/s/raw1' NAME fg1d1
FAILGROUP fg2     DISK '/dev/s/raw2' NAME fg2d2
FAILGROUP fg3     DISK '/dev/s/raw3' NAME fg3d3
FAILGROUP fg4     DISK '/dev/s/raw4' NAME fg4d4 ;
```

View **V$ASM_DISKGROUP** for information on the disk groups. View **V$ASM_OPERATION** to see the status of any ongoing ASM operation.

Drop a diskgroup using the **DROP DISKGROUP** statement:

```
DROP DISKGROUP my_group1 ;
```

Include the keywords **INCLUDING CONTENTS** if you need to drop a diskgroup with data:

```
DROP DISKGROUP my_group1 INCLUDING CONTENTS ;
```

Here are other ASM **ALTER DISKGROUP** commands:

ALTER DISKGROUP Statement Keywords:	**Use:**
...check all	Verifies diskgroup internal consistency
... drop disk	Drops a disk (and auto-rebalances)
... drop ... add	Drops a disk and adds a disk (and auto-rebalances)
...mount	Renders a diskgroup available to client instances
...dismount	Renders a diskgroup unavailable to client instances

ASM File Types and File Naming

ASM supports all Oracle database files except those outside of the database proper (such as the Oracle executables, user traces files, etc).

ASM supports the following files:

- Datafiles
- Control files
- Online Redo logs
- Archived Redo logs
- Temp files
- RMAN files (e.g., datafile backups, datafile copies, incremental backups, etc)
- Flashback logs
- Data Pump datasets
- Others

ASM uses templates to enable it to work with all these different kinds of files. There is a separate template for each file type.

All ASM files are Oracle-Managed Files (or OMF files). There are 6 different ways to refer to a file. It is important to remember the ways to refer to ASM files and when each may be used.

Here are the 6 different kinds of file names:

- **Fully-Qualified File Names:** Used **only** when referencing an existing file.
  Format: +group/dbname/file_type/tag.file.incarnation
  Example: +my_grp/prd/datafile/dataf2.256.1.
- **Numeric Names:** Use **only** when referencing an existing file.
  Example: +my_grp.256.1.
- **Alias Names:** Use to create a file **or** reference an existing file. Create the alias name through the **ALTER DISKGROUP ADD ALIAS** statement.
- **Alias with Template Names:** Use **only** when creating a new file.
- **Incomplete Names:** Use for creating a new file or group of files.
- **Incomplete Names with Template:** Use for creating a new file or group of files.

ASM Initialization Parameters

ASM instances use a number of ASM-unique instance initialization parameters:

Initialization Parm:	**Use:**
INSTANCE_TYPE	Set to **ASM** for an ASM instance (the alternative is the default, **RDMS**, which indicates a database instance).
DB_UNIQUE_NAME	Defaults to **+ASM**. Only set this differently if you're running multiple ASM instances on one node.
ASM_DISKSTRING	Defines what disks will be available to the ASM instance. **NULL** means all disks are visible.
ASM_DISKGROUPS	Lists the diskgroups for ASM to use.
ASM_POWER_LIMIT	How much resource to dedicate to background rebalancing. Ranges from 1 (low) to 11 (high) with default of 1.
LARGE_POOL_SIZE	Must be **8 M or larger.**

New ASM Background Processes

ASM uses several new background processes:

Background Process:	Use:	Runs Where:
RBAL	Coordinates balancing data across disk groups	ASM instance
ARBn	Performs the actual rebalancing of data across disk groups (**n** is 0 thru 9)	ASM instance
ASMB	Communicates to ASM instance	Client instance
RBAL	Performs open/close of disks on behalf of the database	Client instance

Migrating a Database to ASM

Use Recovery Manager (RMAN) to move files from a regular database instance into an ASM instance. Here are the steps:

1. Backup the control file to trace and remember file, control file, and online redo log file names
2. Shutdown Normal or Immediate
3. Backup the database
4. Edit the **SPFILE**
   - Remove **CONTROL_FILES** parameter
   - Use OMF for file destinations
5. Run this script, editing to cover all log files:

```
STARTUP NOMOUNT;
RESTORE  CONTROLFILE  FROM  '<its location>';
ALTER DATABASE MOUNT;
BACKUP AS COPY DATABASE FORMAT
        '<disk group destination>' ;
SWITCH  DATABASE  TO  COPY;
SQL "ALTER DATABASE RENAME <logfile_1>
        TO  '<disk group destination>'  ";
ALTER  DATABASE  OPEN  RESETLOGS ;
```

Storage Enhancements (other than ASM)

10g provides many storage improvements beyond ASM. This section lists and discusses each.

Temporary Tablespace Groups

A temporary tablespace group is a set of one or more temporary tablespaces. It simplifies tablespace management and also helps spread parallel operations.

Temporary tablespace groups may be referenced wherever you would normally reference a temporary tablespace. For example, you can assign a user to a temporary tablespace group instead of a temporary tablespace within the **CREATE USER** statement, just like this:

```
CREATE USER johnny IDENTIFIED BY jc01
DEFAULT TABLESPACE USERS
TEMPORARY TABLESPACE my_ts_group ;
```

Requirements for a temporary tablespace group are:

- It must have at least one member (it cannot exist as empty)
- All members must be *temporary* tablespaces only
- Any temporary tablespace can be a member of only one temporary tablespace group at any one time

The first reference to a temporary tablespace group creates the group.

Here's an example:

```
CREATE TEMPORARY TABLESPACE my_ts
TEMPFILE '/u02/oradata/prd/temp01.dbf'
SIZE 100M
TABLESPACE GROUP my_ts_group ;
```

To take a temporary tablespace out of a temporary tablespace group, issue this statement with the empty single-quotation marks:

```
ALTER TABLESPACE my_ts TABLESPACE GROUP '' ;
```

The new view **DBA_TABLESPACE_GROUPS** lists the temporary tablespace groups and their members.

Renaming Tablespaces

You can now easily rename tablespaces:

```
ALTER TABLESPACE my_ts RENAME my_new_name_ts ;
```

You can **never** rename the **SYSAUX** or **SYSTEM** tablespaces. Nor can you rename any tablespace that is offline or that has one of its datafiles offline.

Renaming a tablespace updates the:

1. Data dictionary
2. Control file
3. Online datafile headers

Renaming a **READ ONLY** tablespace means that the datafile headers are automatically updated when the tablespace is changed to **READ WRITE**.

You should back up the control file as soon as possible after renaming tablespaces. Otherwise, you can have a divergence between the actual tablespace names and the tablespace names in the control file.

Bigfiles

10g introduces bigfile tablespaces. Bigfiles are simple to manage and can grow very large. They can have only a single datafile (no more, "*you cannot add a second datafile to a bigfile tablespace*").

Maximum allowable size for a bigfile depends on the underlying tablespace blocksize:

Tablespace Block Size:	Maximum Bigfile Size:
2 K	8 T
4 K	16 T
8 K	32 T
16 K	64 T
32 K	128 T

So, the maximum size of a bigfile tablespace is 128 Terabytes. With bigfiles, **T** stands for Terabytes while **G** stands for Gigabytes.

Bigfiles **must** be defined as:

- Locally managed
- Automatic segment space management

Default allocation policy is **AUTOALLOCATE**, but it makes sense to change this to **UNIFORM** when you know how big the tablespace will get. (Recall that **UNIFORM** ensures that subsequent space allocations are all the same size.)

By default, all tablespaces created in a 10g database are of type **SMALLFILE** (this is what all tablespaces were prior to 10g). So to create a bigfile tablespace, use the new **BIGFILE** keyword:

```
CREATE BIGFILE TABLESPACE my_bigfile
DATAFILE '/u02/oradata/prd/my_bigfile.dbf'
SIZE 8g AUTOEXTEND ON ;
```

You could change the database default from **SMALLFILE** to **BIGFILE** like this:

```
ALTER DATABASE SET DEFAULT BIGFILE TABLESPACE ;
```

Query dictionary view **DATABASE_PROPERTIES** and its new column named **DEFAULT_TBS_TYPE** to see what the default tablespace type is for your system (it displays as either **SMALLFILE** or **BIGFILE**).

Bigfiles require a new kind of **rowid**.

Here is the traditional rowid:

- **OOOOOO**: Data object number
- **FFF**: Relative datafile number
- **BBBBBB**: Data block (relative)
- **RRR**: Slot number or row number inside a block

The new rowid used for bigfiles are:

- **OOOOOO**: Data object number
- **LLLLLLLLL**: Block number relative to the tablespace
- **RRR**: Slot number or row number inside a block

The **DBMS_ROWID** package is updated with a new parameter, **TS_TYPE_IN,** which is set to either **SMALLFILE** or **BIGFILE**.

Default Permanent Tablespace

You can now define a default permanent tablespace in either the **CREATE DATABASE** or **ALTER DATABASE** statements. It applies to any users not specifically assigned a default permanent tablespace. For example, this statement ensures all new users without other specific assignment will use the new default permanent tablespace **default_ts**:

```
ALTER DATABASE DEFAULT TABLESPACE default_ts ;
```

You cannot drop a default permanent tablespace. You must assign another tablespace in this role, and then you can drop the tablespace previously used in this role. The default permanent tablespace does not apply to system users such as **SYS**, **SYSTEM**, and **OUTLN**.

Index Improvements

Hash-Partitioned Global Indexes — In addition to range-partitioned indexes, 10g now supports hash-partitioned global indexes. Hash-partitioned global indexes offer several advantages including greater parallelism in some situations.

To create a hash-partitioned global index, specify the **BY HASH** keywords:

```
CREATE INDEX hash_ind_ex ON car_tracking_table(sticker_num)
        GLOBAL PARTITION BY HASH(sticker_num)
        (PARTITION p1 TABLESPACE p1_idx,
         PARTITION p2 TABLESPACE p2_idx,
         PARTITION p3 TABLESPACE p3_idx ) ;
```

You can use the **ALTER INDEX ... ADD PARTITION** and **COALESCE PARTITION** clauses with both hash-partitioned and range-partitioned indexes. Recall that these add an index partition and drop an index partition while automatically redistributing its contents, respectively.

Some range-partitioning index operations are not available for hash-partitioned indexes. Examples are **SPLIT INDEX PARTITION** and **MERGE INDEX PARTITION** (obviously these do not make sense in a hash context versus a range index). The only modification you can make with a hash-partitioned index and the **ALTER INDEX MODIFY PARTITION** statement is to mark the partition **UNUSABLE**.

For range-partitioned indexes, the degree of parallelism is limited to the number of index partitions. For hash-partitioned indexes, multiple parallel processes can run against each pruned partition of the global index.

Skipping Unusable Indexes — Prior to 10g, an unusable index partition would result in an ORA-01502 error. Now the initialization parameter **SKIP_UNUSABLE_INDEXES** may be set dynamically at the system or at the session level. Setting this to **TRUE** skips unusable index partitions without generating the ORA- error. The 10g default is **TRUE**. Of course, while the partition and error message are skipped, you may get a sub-optimal execution plan.

When indexes or index partitions become invalid, Oracle puts a message in the Alert Log. You can also view **DBA_IND_PARTITIONS** for this information.

Bitmap Indexes — Bitmap indexes that are subject to many updates deteriorate over time. 10g resists this deterioration better than previous Oracle releases. Set the initialization parameter **COMPATIBLE** to **10.0** or above to get this feature. Then check whether your bitmap index performance deteriorates. If yes, you still may need to rebuild the index, but the number of cases in which this happens are diminished.

Local-Partitioned Index Enhancements — Associated indexes no longer have to reside in the same tablespace as their table. Local-partitioned indexes are now automatically maintained during these operations:

- **ADD PARTITION**
- **SPLIT PARTITION**
- **MERGE PARTITION**
- **MOVE PARTITION**

View **DBA_IND_PARTITION** is the place to look for information about the usability of partitioned indexes.

Partitioned Tables

Tables can be partitioned by these methods:

- Range
- Hash
- List
- Range-Hash
- Range-List

List-partitioned index-oriented tables (IOTs) are now fully supported. IOTs can be partitioned by any column (not just a subset of the primary key columns, as previously).

IOTs can now use local-partitioned bitmap indexes, which requires a mapping table.

All kinds of IOTs now support LOB columns.

Materialized View Enhancements

Materialized views, or MVs, are enhanced in a number of ways. Partition Change Tracking, or PCT, tracks the rows in an MV are changed by update DML. PCT now supports list partitioning as well as range and range-hash partitioning. The new **TRUNCATE REFRESH** operation makes deletion of rows from MVs more efficient.

The procedure **DBMS_MVIEW.EXPLAIN_MVIEW** shows if an MV is fast refreshable or supports query rewrite. The new procedure **DBMS_ADVISOR.TUNE_MVIEW** is part of 10g's new set of Advisors. It advises about MV logs and how to enable fast refreshes and query rewrites. Its results are viewable in **DBA_TUNE_MVIEW**.

The new SQL hint **REWRITE_OR_ERROR** ensures that either a SQL query is rewritten, or that it produces the error ORA-30393 because it could not be rewritten. The new procedure **DBMS_MVIEW.EXPLAIN_REWRITE** explains why a query rewrite failed.

10g supports partition maintenance operations on materialized views. The **ALTER MATERIALIZED VIEW** statement supports:

- Truncating partitions
- Exchanging partitions with a table
- Dropping partitions

Flushing the Buffer Cache

10g allows you to flush the buffer cache with the statement:

```
ALTER SYSTEM FLUSH BUFFER_CACHE ;
```

After you execute this statement, you will have a 100% miss ratio on the cache. The system will not locate anything in the buffer cache as it will (momentarily) be empty. This is useful for setting an identical starting point when testing and comparing re-written SQL statements. Since this causes a performance hit, you'll typically only use it on test (non-production) instances.

Resumable Space Allocation

Resumable space allocation suspends a session when an out-of-space condition occurs in the database. Prior to 10g, you could only turn this on the session level with the statement:

```
ALTER SESSION ENABLE RESUMABLE ;
```

10g allows you to specify resumable space allocation on the database level with the **RESUMABLE_TIMEOUT** initialization parameter. Specify the number of seconds for this parameter to enable database-level resumable space allocation. The default value is 0, so by default, database-level resumable space allocation is disabled.

Check view **DBA_RESUMABLE** to get information on suspended sessions.

The SYSAUX Tablespace

Recall that the new **SYSAUX** tablespace is required for 10g databases. When upgrading databases to 10g, the DBUA tool creates the **SYSAUX** tablespace for you. Or you can create it manually:

```
CREATE TABLESPACE SYSAUX
DATAFILE '/u02/oradata/prd/sysaux.dbf' SIZE 500M
AUTOEXTEND ON
EXTENT MANAGEMENT LOCAL
SEGMENT SPACE MANAGEMENT AUTO ;
```

SYSAUX must be: **PERMANENT**, **ONLINE**, **EXTENT MANAGEMENT LOCAL**, **SEGMENT SPACE MANAGEMENT AUTO**, and read/write. It is a companion tablespace to **SYSTEM**. Both are required, cannot be renamed, and together comprise the minimal required tablespaces for a 10g database.

You also create **SYSAUX** with the **CREATE DATABASE** statement:

```
CREATE DATABASE
DATAFILE '/u02/oradata/prd/system000.dbf' SIZE 500M
SYSAUX DATAFILE '/u02/oradata/prd/sysaux.dbf' SIZE 500M
DEFAULT TEMPORARY TABLESPACE
        TEMP TEMPFILE '/u05/oradata/prd/temp01.dbf' SIZE 200M
UNDO TABLESPACE UNDO01
        DATAFILE '/u05/oradata/prd/undo01.dbf' SIZE 200M ;
```

Note that with 10g, you typically include a default temporary tablespace and undo tablespace on the **CREATE DATABASE** statement.

Dropping a Database

You can drop an entire database with the new **DROP DATABASE** command:

```
DROP DATABASE ;
```

This also deletes all datafiles listed in the control files and the **SPFILE**. The database should be Mounted exclusive and restricted when issuing this command. You require **SYSDBA** privileges to issue the command.

The Recovery Manager (RMAN) also has a new **DROP DATABASE** command. This additionally drops all backup copies and archived log files if you include the **INCLUDING BACKUPS** keywords:

```
RMAN> DROP DATABASE INCLUDING BACKUPS ;
```

Recall that the **UNREGISTER DATABASE** command removes the database information from RMAN's Repository:

```
RMAN> UNREGISTER DATABASE ;
```

Finally, remember that DBCA has the option **Delete a Database** on its main selection panel. This removes the database and its associated files.

Shrinking Segments

10g now allows you to make segments smaller and eliminate internal unused space. You can optionally move the High Water Mark (HWM) down, which would free up space for other segments. This new feature is called segment shrink.

To perform segment shrink, you must first enable row movement because the process can change some ROWIDs:

```
ALTER TABLE table_name ENABLE ROW MOVEMENT ;
```

Then, shrink the segment and move the HWM:

```
ALTER TABLE table_name SHRINK SPACE ;
```

While rows are moved in the shrink operation, small numbers of rows are locked during the compaction process, but the data in the table is fully available to other users. However, the entire table is locked for a very brief period while the HWM is moved down.

If you want to shrink space **without** moving the HWM, issue this statement:

```
ALTER TABLE table_name SHRINK SPACE COMPACT ;
```

Later, you can finish the operation by issuing:

```
ALTER TABLE table_name SHRINK SPACE ;
```

Shrinking without the **COMPACT** keyword invalidates all cursors on the segment.

If you want to shrink all dependent objects (such as indexes), be sure to specify the **CASCADE** keyword:

```
ALTER TABLE table_name SHRINK SPACE CASCADE ;
```

You cannot shrink segments with these characteristics:

- Segments lacking automatic segment space management (in other words, segments using freelists for space management)
- Tables with:
  - LONG columns
  - On-commit or ROWID-based materialized views
  - Function-based indexes
  - Clustered tables
- LOB segments
- IOT mapping tables or overflow segments

Sorted Hash Clusters

Sorted hash clusters ensure that rows will be stored in a hash cluster in the order you specify when creating the cluster. This is the first time Oracle databases allow control over the order in which rows are stored in heap tables.

Sorted hash clusters are useful if your applications retrieve data in a certain order. They eliminate the overhead of sort operations when retrieving the data in this order.

To use them, first create the sorted hash cluster. This example establishes **car_sticker_numb** as the column on which the hash key values are built:

```
CREATE CLUSTER car_sticker_cluster (
        car_sticker_numb NUMBER ,
        area_id                 NUMBER  SORT,
        issue_date              DATE SORT )
HASHKEYS 60
HASH IS car_sticker_numb ;
```

This statement creates the table associated with the cluster:

```
CREATE TABLE car_sticker_detail (
        car_sticker_numb NUMBER,
        area_id                 NUMBER SORT,
        issue_date              DATE SORT,
        tow_lot         NUMBER,
        pk_fee_assign           NUMBER )
CLUSTER car_sticker_cluster (car_sticker_numb, area_id, issue_date) ;
```

Notice the use of the **SORT** keyword in the definitions.

A query like this would not require a sort for retrieval:

```
SELECT * FROM car_sticker_detail ORDER BY area_id ;
```

But a query like this one would require a sort:

```
SELECT * FROM car_sticker_detail ORDER BY tow_lot ;
```

Flash Recovery and RMAN Improvements

The Flashback Recovery Tools

10g Flashback is a set of features designed to enable recovery from logical errors. By logical errors, we mean human error such as running a batch update program twice by mistake or accidentally dropping a table or running an incorrect query. Flashback can sometimes be used for other kinds of recovery, but this is its primary purpose.

While there are a variety of flashback features, all use the common flashback recovery area (or flash recover area). This area can be a filesystem, directory, or ASM disk group. It can be shared by more than one Oracle database. Recovery from the flashback recover area is fast because it is disk storage. This single area backs up all data for flashback recoveries including:

Flashback Area Resident:	**From:**
Flashback logs	Used by Flashback Database to take an entire database back to an earlier point
Control Files	A control file copy is placed in the flashback area when the database is created
Archived Redo Logs	Initialization parameter **LOG_ARCHIVE_DEST_10** automatically points to the flashback area so that the background processes ARCn automatically copy archived redo logs to the flashback area.
SPFILE backups	As generated by Recovery Manager (RMAN)
RMAN backup sets	As generated by RMAN for regular datafile backups
RMAN image copies (datafile copies)	As generated by RMAN when using RMAN's **BACKUP AS COPY** command

The flashback recovery area does **not** include the password file or Oracle binaries.

The directory structure for the flashback recovery area contains a separate directory for each file type.

Subdirectories are then divided by timestamp, so it's easy to see where things are stored.

To set up the flashback recovery area you must set two dynamic initialization parameters:

- **DB_RECOVERY_FILE_DEST_SIZE**: size of the area
- **DB_RECOVERY_FILE_DEST**: where it's located

You **must** specify the first parameter prior to the second. If using Real Application Clusters (RAC), all instances in the cluster must have the same two values for these parameters. Oracle automatically uses Oracle Managed Files (OMF) for file naming in the flash recovery area.

You can manage the flashback recovery area through the EM Database Control GUI Maintenance link. Oracle automatically manages the file space in the flash recovery area. It deletes obsolete files and gives a Warning Message at 85% filled and a Critical Message at 97% filled. These are shown on EM Database Control and logged in the Alert Log.

To back up the flashback recovery area, issue the **RMAN BACKUP... RECOVERY FILES** command:

```
RMAN> BACKUP DEVICE TYPE SBT RECOVERY FILES ;
```

The view **V$RECOVERY_FILE_DEST** provides information about the flash recovery area. This includes its location, space allocation, what files are there, etc. The new **BYTES** column of view **V$BACKUP_PIECE** also indicates how big a file is in the flash recovery area. A very important new column, **IS_RECOVERY_FILE_DEST**, indicates whether a file exists in the flash recovery area. This new column appears in the views:

- **V$BACKUP_PIECE**
- **V$DATAFILE_COPY**
- **V$CONTROLFILE**
- **V$LOGFILE**
- **V$ARCHIVED_LOG**

The 5 kinds of flashback recovery and their purposes are:

Flashback Feature:	Relies on data from:	Purpose:
Flashback Versions Query	Undo Tablespace, Redo logs	Retrieves all versions of table rows between two SCNs or Timestamps
Flashback Transaction Query	Undo Tablespace, Redo logs	Retrieves all table rows affected by a specific Transaction
Flashback Table	Undo Tablespace, Redo logs	Recovers one or more tables to a specific point in time (without using traditional point-in-time recovery)
Flashback Drop	Recycle Bin	Recovers a dropped table (without using traditional point-in-time recovery)
Flashback Database	Flashback Logs	Recovers the entire database to a prior point-in-time

10g enhances 9i's Flashback Query into two forms:

- Flashback Versions Query
- Flashback Transaction Query

Flashback Versions Query — displays all rows in a table between two System Change Numbers (SCNs) or Timestamps. This shows the rows regardless of whether they were inserted, updated, or deleted. The limitations of Flashback Versions Query are that it does not work for:

- Temporary tables
- External tables
- Fixed tables
- Views
- Ranges during which there were structural changes in the table

The **SELECT** statement format you use for Flashback Versions Query is:

```
SELECT [pseudo_columns]... FROM table_name
VERSIONS BETWEEN
{ SCN | TIMESTAMP {expression | MINVALUE} AND
                  {expression | MAXVALUE} }
[ AS OF {SCN | TIMESTAMP expression} ]
          WHERE ...
```

The **MINVALUE** keyword represents the oldest SCN or timestamp, while the **MAXVALUE** keyword represents the most recent. The possible **pseudo_columns** in the above statement can be:

VERSIONS_STARTSCN	SCN when this version of row was created
VERSIONS_STARTTIME	Timestamp when this version of row was created
VERSIONS_ENDSCN	SCN when this version of row was changed or deleted
VERSIONS_ENDTIME	Timestamp when this version of row was changed or deleted
VERSIONS_XID	Transaction ID that created this version of the row
VERSIONS_OPERATION	Type of DML (I = Insert, U = Update, D = Delete)

Flashback Transaction Query — retrieves all table rows affected by a specific transaction. To use it, simply retrieve information from the dictionary view **FLASHBACK_TRANSACTION_QUERY**.

The columns in this view are:

- XID - transaction ID
- START_SCN - SCN for 1st transaction DML
- START_TIMESTAMP -Timestamp for 1st transaction DML
- COMMIT_SCN - SCN when transaction was committed
- COMMIT_TIMESTAMP - Timestamp when transaction was committed
- LOGON_USER - User for the transaction
- UNDO_CHANGE# - The undo SCN
- OPERATION - Type of DML (Insert, Update, Delete, Unknown)
- TABLE_NAME - Name of table involved
- TABLE_OWNER - Owner of that table
- ROW_ID - The row modified
- UNDO_SQL - The SQL statement to undo the DML operation

You need the **SELECT ANY TRANSACTION** privilege to use this facility.

Flashback Table — allows you to recover one or more tables quickly to a point in time (without using traditional recovery methods). To use it, you must first enable row movement on the table(s) you want to flash back:

```
ALTER TABLE schema.table_name ENABLE ROW MOVEMENT ;
```

Then flashback the table. This example goes back 10 minutes:

```
FLASHBACK TABLE schema.table_name TO TIMESTAMP
   systimestamp - INTERVAL '10' MINUTE ;
```

You can re-run this query again using a different interval if you need to. The command acquires exclusive DML locks on the table (obviously!). You cannot use this feature if a structural change has occurred to the table within the time frame you want to work with.

Flashback Drop — recovers a dropped table to some previous point in time before the drop. It uses the recycle bin, a logical structure based on a dictionary table. You can find out what's in the recycle bin by querying view **DBA_RECYCLEBIN** or **USER_RECYCLEBIN.** You can also issue the **SHOW RECYCLEBIN** command from within SQL*Plus. You may also want to query **DBA_CONSTRAINTS** if working with more than one table, to see if there were logical relationships (dependencies) between them.

Only tables defined within locally managed tablespaces are placed into the recycle bin when dropped.

Dependent objects for eligible tables (like its indexes) are also preserved and recovered, except for bitmap join indexes, referential integrity constraints, and materialized view logs.

To recover a table and its dependent objects, use the **FLASHBACK TABLE... TO BEFORE DROP** command:

```
FLASHBACK TABLE table_name TO BEFORE DROP ;
```

You might also want to rename the recovered table in the process:

```
FLASHBACK TABLE table_name TO BEFORE DROP
RENAME TO my_recovered_table ;
```

A big concern with flashback drop is: how far back can you go? In part, this depends on the size of the recycle bin and how it reuses space. Objects in the recycle bin are deleted on a First-In-First-Out (FIFO) basis. Oracle gets free space required to place a newly-dropped object into the recycle bin in this order:

1. Unoccupied free space in the recycle bin
2. Free space taken from dropped objects
3. Autoextension of the recycle bin tablespace

Note that dropped objects are purged (2) prior to autoextending the tablespace (3).

You can manually free space from the recycle bin through the **PURGE TABLE** and **PURGE INDEX** commands. Or use **PURGE TABLESPACE** to get rid of a set of objects.

The **PURGE RECYCLEBIN** (aka **PURGE USER_RECYCLEBIN**) command purges all objects a user owns from the recycle bin, while **PURGE DBA_RECYCLEBIN** purges all objects. The latter requires the **DBA** or **SYSDBA** privilege.

DBA_FREE_SPACE and **DBA_RECYCLEBIN** views are where to look to see what's in the recycle bin and their sizes.

If you want to drop a table without putting it in the recycle bin (basically bypassing the recycle bin), you'll want to use the new PURGE keyword:

```
DROP TABLE table_name PURGE ;
```

This permanently drops the table and its data such that Flashback Drop is not a possibility.

Flashback Database — recovers an entire database to a prior point in time but via a means that is much faster than traditional recovery. Flashback database uses the new background process RVWR (Recovery Writer) to write data from the flashback buffer in the System Global Area (SGA) to the flashback logs in the flash recovery area.

Here are the steps to set up Flashback Database:

1. STARTUP MOUNT EXCLUSIVE the database
2. ALTER SYSTEM SET DB_FLASHBACK_RETENTION_TARGET = n minutes
3. ALTER DATABASE FLASHBACK ON ;
4. ALTER DATABASE OPEN;

You must be in **MOUNT EXCLUSIVE** mode with an Archivelog mode database to enable Flashback Database. Set the **DB_FLASHBACK_RETENTION_TARGET** to the number of minutes for eligible flashback. Changes to the database will now be written both to the redo logs and the flashback logs in the flashback recovery area. Check whether flashback database is enabled via:

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

View **V$FLASHBACK_DATABASE_LOG** helps monitor the estimated size of the flashback logs, while **V$FLASHBACK_DATABASE_STAT** shows how much flashback log is being written per time period.

You can exclude a specific tablespace from a flashback enabled database by:

```
ALTER TABLESPACE my_ts FLASHBACK OFF;
```

Column **FLASHBACK_ON** in **V$TABLESPACE** tells whether flashback is enabled for each tablespace.

Figure 3: Recovering a Database with Flashback Database

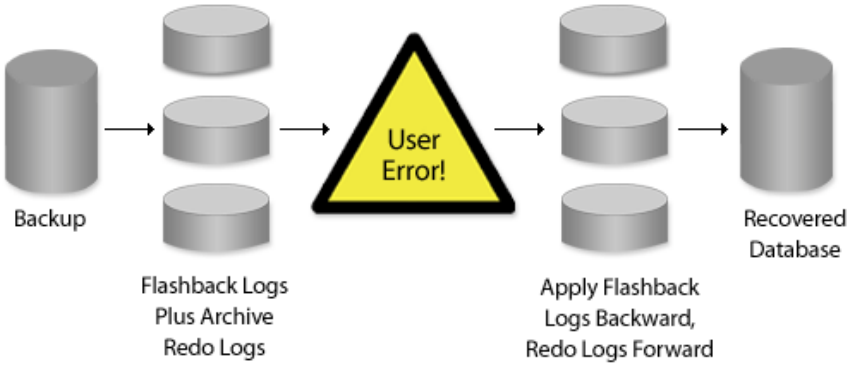

Figure 3 diagrams how recovering a database by flashback database works.

Follow these steps:

1. SHUTDOWN IMMEDIATE;
2. STARTUP MOUNT EXCLUSIVE ;
3. FLASHBACK DATABASE TO TIMESTAMP(some_time) ;
4. ALTER DATABASE OPEN RESETLOGS;

You **must** use **STARTUP MOUNT EXCLUSIVE** and open the database with **RESETLOGS**. You can flash back to a timestamp or an SCN.

Flashback database will not recover tables dropped after the target recovery time. You cannot use flashback database if the control file was restored or re-created after the target recovery time or if the database was opened with **RESETLOGS** after the target recover time.

Recovery Manager (RMAN) Improvements

RMAN now supports incrementally updated backups. This speeds both backups and recoveries (because fewer logs are required in recovery). It applies changes to an existing RMAN image copy. Oracle recommends a strategy of a full database backup, with incremental backups and incrementally updated backups daily.

Fast incremental backups are another new feature in 10g. This creates incremental backups by using a change-tracking file, which tracks changed blocks. So now, incremental backups only need to copy the changed blocks and can ignore the rest for a much faster incremental backup.

The size of the change tracking file is proportional to the size of the database and the number of nodes in a RAC cluster. Change tracking is done by the new background process CTWR (Change Tracking Writer). This file must be 10M or larger.

If you use Oracle-Managed Files (OMF), enable block change tracking like this:

```
ALTER DATABASE ENABLE BLOCK CHANGE TRACKING ;
```

The location of the change tracking file will be dictated by the initialization parameter controlling OMF, **DB_CREATE_FILE_DEST**. If you're not using OMF, specify the **USING FILE** clause to determine the location of the change tracking file:

```
ALTER DATABASE ENABLE BLOCK CHANGE TRACKING
USING FILE '/u05/oradata/prd/change_tracking/chg01.dbf' ;
```

Turn the feature off by:

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

Dynamic view **V$BLOCK_CHANGE_TRACKING** tells where the block change tracking file resides.

You can now drop a database through RMAN:

```
DROP DATABASE INCLUDING BACKUPS ;
```

You can now put all tablespaces into online backup mode for manual backup in a single command:

```
ALTER DATABASE BEGIN BACKUP ;
```

Turn off online backup mode with:

```
ALTER DATABASE END BACKUP ;
```

With the database-wide **BEGIN BACKUP** command, you do not get any error messages if any datafiles are missing, offline or read-only.

You can use this RMAN command to make image copy backups:

```
RMAN> BACKUP AS COPY TABLESPACE my_ts_1, my_ts_2 ;
```

Or backup the entire database as image copies:

```
RMAN> BACKUP AS COPY DATABASE ;
```

To backup the entire database in backup sets, issue:

```
RMAN> BACKUP DATABASE ;
```

To backup a previous copy of the entire database:

```
RMAN> BACKUP COPY OF DATABASE ;
```

Note the subtle difference in syntax between backing up the entire database as an image copy versus backing up the database backup.

To backup the control file to the flash recovery area:

```
RMAN> BACKUP CURRENT CONTROLFILE ;
```

You can also now compress RMAN backup sets, but not RMAN image copies. This reduces the space required for the backup. This command compresses a backup set:

```
RMAN> BACKUP AS COMPRESSED BACKUPSET
        TABLESPACE my_ts ;
```

To make compressed backups to disk the default issue:

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

A really quick new way to recover your entire database uses the flash recovery area and this command:

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

When you issue this command, RMAN resets the location of the datafiles in the control file to point to those residing in the flash recovery area. This is very fast for recover, but the drawback is that your datafiles are now running out of the flash recovery area.

10g improves opening a database with **RESETLOGS** in that log names now contain an incarnation number. So you could still use logs **prior** to issuing this command in recovery if you want:

```
ALTER DATABASE OPEN RESETLOGS
```

Opening a database as **RESETLOGS** no longer obsoletes logs collected prior to the time the **OPEN RESETLOGS** was issued. You do not have to delete or move the old logs. They are still available and usable, if you ever want or need them.

10g gives you greater control over backup times and the load backups impose on the system. The new **DURATION** keyword specifies a time frame for completion of a backup. This example gives the backup a maximum of a 4-hour period:

```
RMAN> BACKUP TABLESPACE my_ts DURATION 4:00 ;
```

You can also minimize the load the backup imposes on the system by **DURATION ... MINIMIZE LOAD**, or complete the backup in the shortest time possible with **DURATION... MINIMIZE TIME**.

This example spreads the backup over a 6-hour period while minimizing the load or performance impact on the database system:

```
RMAN>  BACKUP  TABLESPACE  my_ts  DURATION 6:00
          MINIMIZE LOAD ;
```

Add the optional **PARTIAL** keyword to prevent the **BACKUP** command from issuing an error message if the backup does not complete within the specified time period. Completed backup sets will be usable, but partial or incomplete ones will have to be re-run.

LogMiner Enhancements

Recall that LogMiner extracts information from Redo log files. Now LogMiner no longer requires you to specify an explicit time frame or SCN range to mine. Just specify a starting point and the **CONTINUOUS_MINE** option and LogMiner does the rest. Here's an example:

```
DBMS_LOGMNR.START_LOGMNR(
options   => DBMS_LOGMNR.CONTINUOUS_MINE ,
starttime => to_date('21-SEP-05 11:00', 'DD-MON-YY HH24:MI')  ) ;
```

Data Guard Enhancements

Among the new 10g features for Data Guard are:

- Can apply redo in real-time on both physical and logical standby databases
- The ARCH process writes directly to standby logs
- Can assign threads to standby redo log groups
- New features for transmission of database generated redo (such as optional encryption of the redo stream and the need to set up authentication for all databases)

End-to-End Application Tracing

10g helps you trace an application from end to end. Do this either through the EM GUI or by the **DBMS_MONITOR** package. You can trace either at a global database level or at a session level.
Use these key procedures in **DBMS_MONITOR**:

DBMS_MONITOR Procedure:	**Use:**
CLIENT_ID_TRACE_ENABLE	Enables tracing based on a Client identifier
CLIENT_ID_TRACE_DISABLE	Disables tracing based on a Client identifier
SESSION_TRACE_ENABLE	Enables session tracing (based on **SID** and **SERIAL#** from view **V$SESSION**)
SESSION_TRACE_DISABLE	Disable session tracing

SERV_MOD_ACT_TRACE_ENABLE	Enables tracing as based on Service Name, Module, and Action
SERV_MOD_ACT_TRACE_DISABLE	Disables tracing as based on Service Name, Module, and Action

The view **DBA_ENABLED_TRACES** gives information on the enabled traces.

If you end up with more than one trace file, use the new **trcsess** utility to combine them into a single file.

Then format their output using the traditional **tkprof** utility.

SQL Enhancements

As with any new release, 10g supports a plethora of SQL enhancements.
These include:

- Regular expressions
- An enhanced **MERGE** statement
- Inter-row (spreadsheet-style) calculations with the MODEL statement
- Data type improvements for LOBs and floating point numbers
- Partition Outer Join converts sparse data to dense
- Case- and accent- insensitive sorts
- A new quote operator
- Database connectivity without configuration files
- Many minor SQL*Plus enhancements
- PL/SQL compiler enhancements
- New PL/SQL packages

Regular Expressions

Regular expressions are a way to succinctly express patterns for strings. They are very powerful for text matching and searching.

10g supports 4 new regular expression SQL functions:

Function:	Use:
REGEXP_LIKE	Similar to **LIKE** but with a regular expression; matches a regular expression pattern.
REGEXP_INSTR	Similar to **INSTR** but with a regular expression; matches a regular expression pattern.
REGEXP_SUBSTR	Similar to **SUBSTR** but with a regular expression; finds a substring within a search string using a regular expression.
REGEXP_REPLACE	Similar to **REPLACE** but with a regular expression; replaces a substring within a search string using a regular expression.

Here's an example using **REGEXP_LIKE**:

```
SELECT  first_name, last_name  FROM  employee_table
WHERE   REGEXP_LIKE (last_name, '^[aeiou]', 'i') ;
```

This statement uses a regular expression to find employees whose last name starts with any vowel. The caret (**^**) represents the beginning of the line, while the pattern **[aeiou]** matches any single character that is a vowel. The **'i'** parameter dictates a case-insensitive pattern match (or search).

The Enhanced MERGE Statement

The **MERGE** SQL statement features these new 10g improvements:

- Either the **UPDATE** or **INSERT** can now be omitted (Previously both had to be specified)
- Conditional **UPDATE**s and **INSERT**s are allowed (Add an optional **WHERE** clause to skip an **UPDATE** or **INSERT** for rows that meet the criteria)
- The new **ON** keyword allows inserting all rows to the target without joining the source and target tables
- The new optional **DELETE** keyword removes rows after the UPDATE has processed

Here is an example of a conditional update **MERGE**. The **WHEN MATCHED** clause only acts when the **WHERE** condition is true:

```
MERGE INTO car_history  ch
USING car_history_adds cha
ON (ch.vehno = cha.vehno)
WHEN MATCHED THEN
UPDATE SET ch.total = ch.total + cha.addl_chge
WHERE ch.type_code = 'EXPENSES' ;
```

Here is an example of the new **ON** condition. It inserts all rows in the source table into the target table (without requiring you to join those two tables):

```
MERGE INTO new_cars  nc
USING cars  c
ON (1=0)
WHEN NOT MATCHED THEN
INSERT (ns.vehno, ns.type_code, ns.new_total)
VALUES  (ns.vehno, ns.type_code, ns.total)
```

MODEL Clause for SELECT Statements

The **MODEL** clause of the **SELECT** statement defines a multidimensional array similar to a spreadsheet. It maps the columns of a query into 3 groups:

- **Partitions** - logical blocks of the result set
- **Dimensions** - cells within a partition
- **Measures** - the data cells

The **MODEL** clause does not perform updates; it merely changes the representation of the resultset coming back from a **SELECT** statement. Code the **MODEL** clause after the **WHERE** and **GROUP BY** clauses but before any **ORDER BY**.

The **MODEL** clause gets more complicated than you need to know for the test, so we won't describe it further here. It includes positional and symbolic cell references, a **FOR** loop, and ways to order rules and ignore nulls.

Data Type Enhancements

10g LOB column size has increased from a maximum of 4G to a maximum that ranges from 8T to 128T depending on the block size. This is rather similar to how the maximum size of Bigfiles range these same limits depending on the block size (see the table that relates block size to the maximum size of a bigfile in the section labeled "Bigfiles" above). The LOB columns include CLOB, BLOB, and NCLOB data types. LOBs now also work like any other data type for the **:NEW** attribute in database triggers.

10g introduces 2 new data types. Both are fully supported with Oracle's SQL and SQL*Plus:

Data Type:	Use:	Bytes Required for Internal Storage:
BINARY_FLOAT	Single precision 32-bit binary	5
BINARY_DOUBLE	Double precision 64-bit binary	9

This example creates a table using the new binary data types:

```
CREATE TABLE my_binary_table (
binary_32_bit          BINARY_FLOAT,
binary_64_bit          BINARY_DOUBLE ) ;
```

Insert values into the table using a simple notation with **f** or **d** as the suffix:

```
INSERT INTO my_binary_table VALUES (2.0f, 9.0d) ;
```

Partitioned Outer Join

A partitioned outer join can be used to convert sparse data into dense data. It is useful in time period manipulations because if no value exists for a specific column position, no row exists in the fact table. It makes time series calculations easier because dense data fill a consistent number of rows for each time period.

A partitioned outer join extends the regular ANSI outer join syntax to apply the outer join to each partition defined in the query. It is also known as the group outer join. It supports the **RIGHT OUTER JOIN** and the **LEFT OUTER JOIN**, but not the **FULL OUTER JOIN**.

Here is the basic format for a **PARTITION BY ... RIGHT OUTER JOIN**:

```
SELECT select_expression
FROM   table_reference
PARTITION BY (expr_list ... )
RIGHT OUTER JOIN table_reference ;
```

The format is similar for the **PARTITION BY ... LEFT OUTER JOIN**. Just replace keyword **RIGHT** with keyword **LEFT**.

Case- and Accent- Insensitive Sorts

The **NLS_SORT** parameter tells Oracle how to perform sorts. You can dictate case-insensitive searches by appending the letters **_CI** to **NLS_SORT**, and you can direct accent-insensitive searches by appending the letters **_AI** to **NLS_SORT**.

The SQL clauses that support the **NLS_SORT** settings include **WHERE**, **ORDER BY**, **HAVING**, **IN**, **NOT IN**, **BETWEEN**, **CASE...WHEN**, and **START WITH**.

New Quote Operator

Prior to 10g, the way to specify a quote mark within a string was to encode two of them, back-to-back. 10g allows you to define your own quote character using the quote operator **q**. Here's the traditional way of coding a quote mark within a character string:

```
SELECT 'Don't tell me what to do!' from dual ;
```

Yields:

```
Don't tell me what to do!
```

Now you can code this query this way and get the exact same result:

```
SELECT q'<Don't tell me what to do!>' from dual;
```

Database Connectivity Without Configuration Files

For TCP/IP networks only, you can now connect to Oracle without a configuration file by fully specifying the connect string in this format:

```
CONNECT username/password@//hostname [:port_number] [/service_name]
```

The hostname is the sole require parameter. Here is an example:

```
CONNECT mylogin/my_pass@//my_host:1521/PROD0001
```

Advanced features such as load balancing and connect time failover do not work when connecting in this manner.

Many Minor SQL*Plus Enhancements

You can now use the **APPEND** keyword when spooling in SQL*Plus, thereby appending new output to an existing file:

```
SQL> SPOOL my_file.txt APPEND
```

You can embed blanks or "white space" within filenames. This is useful for Windows platforms:

```
SQL> "C:\Program Files\my report.txt"
```

You can view the objects in the Recyclebin for the Flashback feature:

```
SQL> SHOW RECYCLEBIN
```

You used to have to specify quotes when invoking SQL*Plus on some platforms:

```
C:> sqlplus '/ as sysdba'
```

Now you can omit the quotes:

```
C:> sqlplus / as sysdba
```

You can set the **SQLPLUSCOMPATIBILITY** variable when invoking SQL*Plus to ensure a compatibility level by using the **-c** option:

```
C:> sqlplus -c 9.2 / as sysdba
```

SQL*Plus offers several new predefined variables:

New Predefined Variable:	Use:
_CLIENT_IDENTIFIER	Connection identifier for the database connection
_DATE	Current date
_EDITOR	Default editor
_O_VERSION	Oracle database version
_O_RELEASE	Oracle database full release number
_PRIVILEGE	Privilege level of the connection
_USER	User name for the connection

PL/SQL Compiler Enhancements

10g's PL/SQL compiler is re-written for speed and code optimization. Control it through these 3 new initialization parameters:

New Compiler Option:	Use:
PLSQL_DEBUG	Directs the PL/SQL library units to be compiled for debugging if **TRUE**. Default is **FALSE**.
PLSQL_OPTIMIZE_LEVEL	Tells how much time the optimizer should spend optimizing. Valid settings are 1 and 2, with 2 being the highest optimization. Default is 2.
PLSQL_CODE_TYPE	Specifies if code is **NATIVE** or **INTERPRETED**. **NATIVE** runs faster, **INTERPRETED** is the default.

New PL/SQL packages

The most important, new, and enhanced 10g packages have been discussed throughout this Exam Manual. Be familiar with these packages and their procedures for the exam:

Package:	Use:
DBMS_ADVISOR	Manage all the new Advisors
DBMS_DATAPUMP	Manage Data Pump for moving data and metadata
DBMS_FILE_TRANSFER	Copy binary files
DBMS_SCHEDULER	Manage the new Scheduler
DBMS_SERVER_ALERT	Set/get alert threshold values
DBMS_SQLTUNE	Manage the SQL Tuning Advisor
DBMS_WORKLOAD_REPOSITORY	Manage the AWR (the statistics repository)
UTL_COMPRESS	Data compression utilities for RAW, BLOB, and BFILEdata types
UTL_MAIL	Utilities to manage email

Resource Manager and Security Enhancements

Recall that the Resource Manager gives you greater control over allocation of resources. **DBMS_RESOURCE_MANAGER** and **DBMS_RESOURCE_MANAGER_PRIVS** are the two packages by which you manage the Resource Manager. The EM Database Control Administration tab provides equivalent GUI control.

The **DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE** and **UPDATE_PLAN_DIRECTIVE** procedures have the **SWITCH_TIME** parameter, which switches a session to a new resource group after a certain number of seconds has passed. The new **SWITCH_TIME_IN_CALL** parameter is also specified in seconds and is mutually exclusive to **SWITCH_TIME**. **SWITCH_TIME_IN_CALL** specifies that at the end of a switched resource group call, the session returns to its original resource group.

Both **SWITCH_TIME** and **SWITCH_TIME_IN_CALL** are specified in seconds and have a default value of **NULL.**

Here's an example of using **SWITCH_TIME_IN_CALL:**

```
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(
        PLAN                        => 'PLAN_ONE',
        GROUP_OR_SUBPLAN    => 'GRP_ONE',
        CPU_P1                      => 100,
        SWITCH_GROUP            => 'QUERIES_GRP',
        SWITCH_ESTIMATE => TRUE,
        SWITCH_TIME_IN_CALL => 30 ) ;
```

This example might be useful for multi-tiered applications that create a pool of sessions that clients share. Previously, after the consumer group changed, all subsequent connections were penalized. In this example, long-running sessions switch to the group **QUERIES_GRP**, but the **SWITCH_TIME_IN_CALL** parameter reverts back to the original plan after completing long-running operations.

Two new 10g settings limit the idle time for a session:

- **MAX_IDLE_TIME** - maximum allowable idle time for a session
- **MAX_IDLE_BLOCKER_TIME**- maximum time a session can be idle and blocking another

Both parameters are specified in seconds and have a default value of **NULL** (unlimited time).

Here is an example showing their use. This creates a plan directive that allows a session to sit idle for a maximum of 200 seconds:

```
DBMS_RESOURCE_MANAGER.CREATE_PLAN_DIRECTIVE(
PLAN                        => 'EX_PLAN',
GROUP_OR_SUBPLAN    => 'EX_GROUP',
MAX_IDLE_TIME           => 200,
COMMENT                     => 'Set MAX_IDLE_TIME' ) ;
```

This example says a session can only be idle for 360 seconds, and can only be idle while blocking another session from acquiring resources for 60 seconds:

```
DBMS_RESOURCE_MANAGER.UPDATE_PLAN_DIRECTIVE(
        PLAN                          => 'MY_PLAN',
        GROUP_OR_SUBPLAN              => 'MY_PLAN_GROUP',
        NEW_MAX_IDLE_TIME             => 360,
        NEW_MAX_IDLE_BLOCKER_TIME     => 60) ;
```

CPU Allocation

Prior to 10g, the only CPU allocation method was **EMPHASIS**. This is still the default, but now you can also specify the **RATIO** policy, which allocates CPU resources using ratios. Specify **RATIO** as the value for the **CPU_MTH** parameter in the **CREATE_PLAN** procedure to specify ratio-based CPU allocation.

In the **CREATE_CONSUMER_GROUP** procedure, 9i only allowed **CPU_MTH** to be **ROUND-ROBIN**. **ROUND-ROBIN** remains the default but **CPU_MTH** can also be defined as **RUN-TO-COMPLETION**. **ROUND-ROBIN** allocates CPU evenly via a round-robin allocation algorithm whereas **RUN-TO-COMPLETION** gives CPU to the session with the largest active time in order to complete it as soon as possible.

Mapping

The new procedure **DBMS_RESOURCE_MANAGER.SET_CONSUMER_GROUP_MAPPING** can add, modify, or delete entities that map sessions to resource consumer groups based on the session's login and run-time attributes. The attributes include:

- Username
- Service name
- Client OS username
- Client program name
- Client machine
- Module name
- Module name action

The **SET_INITIAL_CONSUMER_GROUP** procedure has been deprecated. Use **SET_CONSUMER_GROUP_MAPPING** instead.

Virtual Private Database (VPD) Enhancements

Column-level Security — Recall that Virtual Private Databases or VPD enables row-level security. You create policies that apply a **WHERE** clause to any **SELECT** statements based on the login ID of the user. The result is that only authorized users view certain rows of VPD-enabled tables.

In 10g, VPD security now can be applied on the column-level as well. The new parameter **SEC_RELEVANT_COLS** in the **DBMS_RLS.ADD_POLICY** procedure ensures that only authorized users can view the columns named in the **SEC_RELEVANT_COLS** parameter. Columns that are masked-out due to security policies show up as blank in any report run by a non-authorized user.

You create a VPD policy just like you did in Oracle 9i. The difference is if you add the new **SEC_RELEVANT_COLS** parameter, in which case the policy applies on the columns you name in that parameter.

Here is an example that applies to the 2 columns **car_model** and **car_make**:

```
DBMS_RLS.ADD_POLICY (
        object_schema     => 'cars_schema' ,
        object_name       => 'cars',
        policy_name       => 'cars_policy' ,
        function_schema   => 'cars' ,
        policy_function   => 'func_cars' ,
        statement_types   => 'SELECT' ,
        sec_relevant_cols => ' car_model, car_make' ) ;
```

New Policy Types — Prior to 10g there was only one policy type: dynamic**.** Dynamic policies are flexible because they execute for each DML statement. But in some cases, this unnecessarily wastes resources. While dynamic remains the default, 10g also offers these other kinds of policies:

- **STATIC**: predicate is assumed the same regardless of the runtime environment
- **SHARED_STATIC**: allows the sharing of Static policies
- **CONTEXT_SENSITIVE**: re-evaluated if the context has changed
- **SHARED_CONTEXT_SENSITIVE**: allows sharing of Context-sensitive policies

Use the **DBMS_RLS** package parameter **POLICY_TYPE** to set these policy types.

Auditing Enhancements

Recall that you enable Oracle's Standard Auditing by setting initialization parameter **AUDIT_TRAIL** on. You specify the objects and type of actions to audit, and audit information is written to the **SYS.AUD$** table and can be seen via the view **DBA_AUDIT_OBJECT**.

Fine-grained auditing, or FGA, is an alternative approach. It used to support only **SELECT** statements, but now it also applies to **INSERT**, **UPDATE**, and **DELETE** statements. You list which DML statements the policy applies to in the **STATEMENT_TYPES** parameter.

The **ANY_COLUMNS** parameter means that a reference to any of the columns listed in the policy will write the FGA record. **ALL_COLUMNS** means that all the columns listed in the policy must be referenced to trigger the FGA record.

The audit row is written during the DML statement. It does not require a **COMMIT**, and will be present even if the qualified DML statement was rolled back.

Let's review a couple examples. This example writes an audit record only if all the columns listed (both **ENAME** and **EMPNO**) are referenced in any **SELECT** or **UPDATE** SQL statement. The audit record populates both the **LSQLTEXT** and **LSQLBIND** columns of the **SYS.FGA_LOG$** table (due to the parameter **DBMS_FGA.DB_EXTENDED**, which is the default):

```
DBMS_FGA.ADD_POLICY(
        policy_name             => 'AUDIT_BOTH_COLS',
        object_schema           => 'HR',
        object_name             => 'EMPLOYEE',
        audit_column            => 'ENAME, EMPNO',
        statement_types         => 'SELECT, UPDATE',
        audit_column_opts       => DBMS_FGA.ALL_COLUMNS,
        audit_trail             =>  DBMS_FGA.DB_EXTENDED );
```

This example policy writes an audit record whenever any of the four basic DML statements executes that meets the audit condition:

```
DBMS_FGA.ADD_POLICY(
        policy_name             => 'AUDIT_COND_EXAMPLE',
        object_schema           => 'HR',
        object_name             => 'EMPLOYEE',
        audit_column            => 'ENAME, EMPNO',
        audit_condition         => 'EMPNO > 55555',
        statement_types         => 'SELECT, UPDATE, INSERT, DELETE' ) ;
```

Key dictionary views for auditing:

SYS.FGA_LOG$	Fine-grained Audit log table (stored in the **SYSTEM** tablespace)
DBA_FGA_AUDIT_TRAIL	The FGA audit trail entries
DBA_AUDIT_TRAIL	Audit trail entries for Standard Auditing
DBA_COMMON_AUDIT_TRAIL	Displays audit records for **both** FGA and Standard Auditing (Combines the entries of DBA_AUDIT_TRAIL and DBA_FGA_AUDIT_TRAIL)
DBA_AUDIT_OBJECT	Shows audit trail records for all objects in the database -- both DML and DDL audit
DBA_AUDIT_POLICIES	Displays the FGA audit policies

Use package **DBMS_FGA** and its procedure **ADD_POLICY** to create the FGA Policy.

The FGA audit trail parameter **AUDIT_TRAIL** defaults to **DBMS_FGA.DB_EXTENDED** — this writes the SQL statement to the column **LSQLTEXT**. Thus, FGA allows you to collect the entire SQL DML in the audit trail based on data access (this distinguishes it from the other auditing alternatives). You can turn off extended logging by specifying **DBMS_FGA.DB**. The default is **DBMS_FGA.DB_EXTENDED.**

An important feature of 10g is the integration of Standard and FGA auditing records to a single view:

DBA_COMMON_AUDIT_TRAIL.

Practice Questions

Chapter 1 Installation

1. You have an Oracle 8.0.5 database and want to upgrade it to release 10g. What is the best way to accomplish this?

   Select the best answer.

   - ❍ A. Perform a direct upgrade to 10g using DBUA.
   - ❍ B. Perform a direct upgrade to 10g manually, by running scripts.
   - ❍ C. Upgrade the database to release 8.0.6 using the methods suggested in that release, then directly upgrade the 8.0.6 database to Oracle 10g using the DBUA.
   - ❍ D. You cannot upgrade an 8.0.5 database to Oracle 10g.
   - ❍ E. Use the unload/load utilities to move the 8.0.5 data into a new Oracle 10g database.

2. You have decided to manually upgrade a database to release 10g. Which answer below places these manual upgrade steps into the proper sequence? (Note -not every database upgrade step is listed.)

   Select the best answer.

   1. Run utlu101s.sql
   2. Run utlrp.sql
   3. Run utlu101i.sql
   4. STARTUP UPGRADE the database
   5. Create the SYSAUX tablespace
   6. Run the database upgrade script appropriate to your release

   - ❍ A. 1,2,3,4,6,5
   - ❍ B. 3,4,5,6,1,2
   - ❍ C. 3,4,5,1,2,6
   - ❍ D. 4,3,5,1,2,6
   - ❍ E. You cannot place these steps into an appropriate sequence because the three scripts referred to as steps 1, 2, and 3 are automatically run for the user during the upgrade after starting the database in STARTUP UPGRADE mode.

3. Which are valid values for the COMPATIBLE parameter when manually upgrading an Oracle 8.1.7 database to 10g?

   Select the best two answers.

   A. 8.1.7
   B. 9.2.0
   C. Any Oracle 8i or 9i value
   D. The COMPATIBLE parameter's default value
   E. You should never tamper with the COMPATIBLE parameter. Just leave it "as is" on the older database to upgrade and the upgrade will work fine.
   F. Just set COMPATIBLE to 10.0.0 regardless, and try it. 10g allows you to downgrade the value of COMPATIBLE to any older release value later.

   ❑ A. A
   ❑ B. B
   ❑ C. C
   ❑ D. D
   ❑ E. E
   ❑ F. F

Chapter 2 Load and Unload Data

1. Select the code snippet that unloads both data and metadata for user SCOTT. The output should include the DDL to create all tables except those that start with the letters TEMP.

   Select the best answer.

   ❍ A. CONTENT=BOTH SCHEMAS=SCOTT EXCLUDE=TABLE:"LIKE 'TEMP%'"
   ❍ B. SCHEMAS=SCOTT EXCLUDE=TABLE:"LIKE 'TEMP%'"
   ❍ C. SCHEMAS=SCOTT INCLUDE=TABLES: EXCEPT "LIKE 'TEMP%'"
   ❍ D. CONTENT=ALL SCHEMAS=SCOTT INCLUDE=TABLES: EXCEPT "LIKE 'TEMP%'"

2. What is the proper sequence to transport tablespaces across different platforms?

   Select the best answer.

   1. Make the tablespaces to transport read-only in the source database
   2. Determine the endian formats of the two platforms
   3. Unload the metadata for the tablespaces to transport using the EXPDP utility
   4. Ensure the tablespaces to be transported are self-contained
   5. Make the tablespaces read-write in the target database
   6. Convert the endian formats -if they are different -using RMAN
   7. Copy the converted datafiles and metadata to the target server, and use the IMPDP utility to import the tablespaces and plug in their metadata to the target server

   ❍ A. 2,4,1,3,6,5,7
   ❍ B. 2,4,1,3,6,7,5
   ❍ C. 6,4,1,3,2,7,5
   ❍ D. 2,4,5,3,6,7,1

Chapter 3 Automatic Management

1. Setting the SGA_TARGET initialization parameter to a non-zero value and STATISTICS_LEVEL to either TYPICAL or ALL automatically distributes memory among which areas?

   Select the best answer.

   - ❍ A. Shared Pool, Large Pool, Java Pool, and the Database Buffer Cache (DB_CACHE_SIZE)
   - ❍ B. Shared Pool, Large Pool, Java Pool, and all Database Buffer Caches (DB_CACHE_SIZE and DB_nK_CACHE_SIZE)
   - ❍ C. Shared Pool, Large Pool, Java Pool, the Log Buffer, the Streams Pool, and all the Database Buffer Caches (DB_CACHE_SIZE and DB_nK_CACHE_SIZE)
   - ❍ D. Log Buffer, all Database Buffer Caches, Streams Pool, and the Fixed-SGA area and internal allocation
   - ❍ E. All memory areas controlled by Oracle
   - ❍ F. No memory areas at all, unless you also configure parameter ASMM=Y

2. Which of the following statements are true?

   Select the three best answers.

   A. When you create a new 10g database using DBUA, the GATHER_STATS_JOB is created and scheduled to run

   B. When you create a new 10g database using DBUA, the GATHER_STATS_JOB is created but not scheduled to run

   C. When you upgrade an existing database to Oracle 10g using DBUA, the GATHER_STATS_JOB is created and scheduled to run

   D. When you upgrade an existing database to Oracle 10g using DBUA, the GATHER_STATS_JOB is created but not scheduled to run

   E. When you upgrade an existing database to Oracle 10g using the manual upgrade, the GATHER_STATS_JOB is created and scheduled to run

   F. When you upgrade an existing database to Oracle 10g using the manual upgrade, the GATHER_STATS_JOB is created but not scheduled to run

   - ❑ A. A
   - ❑ B. B
   - ❑ C. C
   - ❑ D. D
   - ❑ E. E
   - ❑ F. F

3. Which is the proper sequence to successfully migrate a database to use automatic storage management (ASM)?

   Select the best answer.

   1. Delete or archive the source database files
   2. Shut down the source database
   3. Edit the SPFILE to use Oracle Managed Files (OMF) for files and remove the CONTROL_FILES parameter
   4. Make a full cold backup of the source database
   5. Run an RMAN script that moves database objects from non-ASM locations into an ASM disk group
   6. Start up and test the database using ASM

   ❍ A. 2,4,3,5,6,1
   ❍ B. 1,2,3,4,5,6
   ❍ C. 4,2,3,5,1,6
   ❍ D. 2,4,3,5,1,6

Chapter 4 Application Tuning

1. Which two OPTIMIZER_MODEs are not valid in 10g?

   Select the two best answers.

   A. FIRST_ROWS
   B. FIRST_ROWS_n
   C. RULE
   D. ALL_ROWS
   E. CHOOSE

   ❑ A. A
   ❑ B. B
   ❑ C. C
   ❑ D. D
   ❑ E. E

2. You cannot collect statistics for which kinds of tables in 10g?

   Select the best answer.

   ❍ A. Data dictionary tables
   ❍ B. Fixed tables
   ❍ C. User-created tables in non-SYSTEM schemas
   ❍ D. None of the above. 10g allows you to collect statistics on data dictionary, fixed, and user tables

3. Which of the following statements are true about the new initialization parameter SQLTUNE_CATEGORY?

   Select the two best answers.

   - ❍ A. Its default value is DEFAULT.
   - ❍ B. SQLTUNE_CATEGORY is now obsolete in 10g because it has been superseded by SQL-TUNE_COLLECTIONS.
   - ❍ C. SQLTUNE_CATEGORY specifies the category to use when applying a SQL profile to a SQL statement.
   - ❍ D. SQLTUNE_CATEGORY is the part of the Database Resource Manager that prioritizes and allocates CPU (but not I/0) resources to resource plans.
   - ❍ E. SQLTUNE_CATEGORY does not support identical SQL statements in multiple profiles.

Chapter 5 System Resource Management

1. Which of the following statements are true about Resource Manager enhancements in Oracle 10g?

   Select the two best answers.

   A. The SWITCH_TIME_IN_CALL parameter obsoletes and replaces the SWITCH_TIME parameter

   B. The SWITCH_TIME_IN_CALL and SWITCH_TIME parameters can be coded together to manage session resources in a single resource group

   C. The SWITCH_TIME_IN_CALL parameter is mutually exclusive to the SWITCH_TIME parameter for a single resource group

   D. The SWITCH_TIME parameter permanently switches a session to a new resource group for the duration of a session, whereas SWITCH_TIME_IN_CALL eventually returns a session to its original resource group

   - ❑ A. A
   - ❑ B. B
   - ❑ C. C
   - ❑ D. D

Chapter 6 Automating Tasks with the Scheduler

1. Which of the following are advantages of the DBMS_SCHEDULER package over the older DBMS_JOB package?

   Select the best two answers.

   A. DBMS_SCHEDULER has several components that interact to provide scheduling capabilities
   B. Job Classes are no longer used to prioritize jobs
   C. DBMS_JOB is no longer available at all in 10g
   D. Like DBMS_JOB, DBMS_SCHEDULER executes only stored programs and anonymous blocks
   E. DBMS_SCHEDULER can use advanced calendaring expressions

   ❑ A. A
   ❑ B. B
   ❑ C. C
   ❑ D. D
   ❑ E. E

Chapter 7 Space Management

1. You run the following two statements on the large EMPLOYEES table:

   SQL> alter table prod.employees enable row movement ;
   SQL> alter table prod.employees shrink space ;

   Based on these statements, which of the following answers is not true?

   Select the best answer.

   ❍ A. Space below the High Water Mark is freed up.
   ❍ B. Full table scans on the table will now take less time.
   ❍ C. All chained rows are fixed.
   ❍ D. The High Water Mark is moved down.
   ❍ E. Data blocks are now contiguous, and fragmented wasted space is reclaimed.

2. Which of the following statements about sorted hash clusters is not true?

   Select the best answer.

   ❍ A. One or more tables can be stored in sorted hash clusters.
   ❍ B. Rows within a given cluster key value are sorted by sort key.
   ❍ C. Cluster key values are hashed.
   ❍ D. The ORDER BY clause is not necessary if you are retrieving rows for a single hash cluster key and want to order rows by the sort key columns.
   ❍ E. You cannot create indexes on sorted hash clusters.

Chapter 8 Improved VLDB Support

1. Which of the following statements can create a temporary tablespace group and assign it one member?

   Select the two best answers.

   A. create temporary tablespace tempmember1 tempfile 'tempmember1.dbf' size 500m tablespace group tempgroup1 ;

   B. create temporary tablespace tempmember1 tempfile 'tempmember1.dbf' size 500m add to tablespace group tempgroup1 ;

   C. create temporary tablespace tempmember1 tempfile 'tempmember1.dbf' size 500m tablespace group '' ;

   D. create temporary tablespace group tempgroup1 add member tempmember1 tempfile 'tempmember.dbf' size 500m ;

   E. alter tablespace tempmember1 tablespace group tempgroup1 ;

   ❑ A. A
   ❑ B. B
   ❑ C. C
   ❑ D. D
   ❑ E. E

Chapter 9 Backup and Recovery Enhancements

1. Which two statements about fast incremental backups are true?

   Select the two best answers.

   ❑ A. The change block tracking file must be 10 megabytes in size or larger.
   ❑ B. Fast incremental backups use a change block tracking file and a new background process called FIBW (Fast Incremental Backup Writer).
   ❑ C. The fast incremental backup feature and its change block tracking file is enabled by default.
   ❑ D. To enable or disable fast incremental backups and the change block tracking feature, you must always stop and restart the instance.
   ❑ E. Oracle implements fast incremental backups by using a special disk device driver.
   ❑ F. Fast incremental backups use a change block tracking file and a new background process called CTWR (Change Tracking Writer).

Chapter 10 Flashback any Error

1. To set up the flash recovery area for a 10g database, which two parameters do you define and in what order must they be defined?

   Select the best answer.

   - ❍ A. DB_RECOVERY_FILE_DEST then DB_RECOVERY_FILE_DEST_SIZE
   - ❍ B. DB_RECOVERY_FILE_DEST_SIZE then DB_RECOVERY_FILE_DEST
   - ❍ C. DB_RECOVERY_FILE then DB_RECOVERY_FILE_SIZE
   - ❍ D. DB_RECOVERY_FILE_SIZE then DB_RECOVERY_FILE

2. Which two statements about the new flashback database feature are true?

   Select the two best answers.

   - ❑ A. Flashback database allows you to "flash back" selected tablespaces within the database to previous points in time that you specify.
   - ❑ B. You can inspect view V$DATABASE to see if flashback database is enabled.
   - ❑ C. When flashback database is enabled, the new background process RVWR runs and writes changed data from the flashback buffer in the SGA to the flashback logs in the flash recovery area.
   - ❑ D. The new initialization parameter DB_FLASHBACK_RETENTION_TARGET should be set to the number of seconds you want to be able to flash back the database into the past.
   - ❑ E. When you flashback a database to a prior point in time, Oracle uses the Undo tablespace data to go back in time, then the Redo logs to roll forward in time to a consistent transactional state.

Chapter 11 General Storage Management

1. The SYSAUX tablespace is required in Oracle 10g. What parameters are necessary for a SYSAUX tablespace?

   Select the best answer.

   - ❍ A. ONLINE or OFFLINE, PERMANENT, READ WRITE, EXTENT MANAGEMENT LOCAL, SEGMENT SPACE MANAGEMENT AUTO
   - ❍ B. ONLINE, PERMANENT, READ WRITE, EXTENT MANAGEMENT LOCAL, SEGMENT SPACE MANAGEMENT AUTO
   - ❍ C. ONLINE, TEMPORARY, READ WRITE, EXTENT MANAGEMENT LOCAL, SEGMENT SPACE MANAGEMENT AUTO
   - ❍ D. ONLINE, PERMANENT, READ ONLY, EXTENT MANAGEMENT LOCAL, SEGMENT SPACE MANAGEMENT AUTO
   - ❍ E. ONLINE, TEMPORARY, READ ONLY, EXTENT MANAGEMENT LOCAL, SEGMENT SPACE MANAGEMENT AUTO
   - ❍ F. ONLINE, PERMANENT, READ WRITE, EXTENT MANAGEMENT LOCAL

2. How do you move an occupant out of the SYSAUX tablespace and into an alternative location?

   Select the best answer.

   - ❍ A. Create a new smallfile tablespace, and then run DBMS_SYSAUX.MOVE_PROCEDURE to relocate the occupant to the new smallfile tablespace.
   - ❍ B. Run this command: SELECT OCCUPANT_NAME, MOVE_PROCEDURE FROM V$SYSAUX_OCCUPANTS ; Then, use the retrieved move_procedure to relocate the occupant.
   - ❍ C. Create a smallfile tablespace, shutdown the database, then use the new COPY_FILE procedure to relocate the occupant.
   - ❍ D. You cannot move occupants out of the SYSAUX tablespace. It is essential to keep the SYSAUX tablespace ONLINE for the database to function.

Chapter 12 Security

1. Which answer is true concerning the following statement? SQL> DBMS_FGA.ADD_POLICY(policy_name => 'MY_HR_POLICY', object_schema => 'HR', object_name => 'EMPLOYEE', audit column => 'FIRST_NAME, LAST_NAME', statement_types => 'UPDATE, INSERT, DELETE', audit_column_opts => DBMS_FGA.ALL_COLUMNS, audit_trail => DBMS_FGA.DB) ;

   Select the best answer.

   - ❍ A. The policy audits all SQL statements that access both the FIRST_NAME and LAST_NAME columns.
   - ❍ B. The policy audits all SQL statements that access either the FIRST_NAME and LAST_NAME columns.
   - ❍ C. The policy audits all UPDATE, INSERT, and DELETE statements that access either the FIRST_NAME and LAST_NAME columns.
   - ❍ D. The policy audits all UPDATE, INSERT, and DELETE statements that access both the FIRST_NAME and LAST_NAME columns.

Chapter 13 Miscellaneous New Features

1. How can you get accent-insensitive and case-insensitive queries and sorts?

   Select the best answer.

   - ❍ A. Append the letters _AI to the NLS_SORT and NLS_SEARCH parameters for accent-insensitive queries and sorts, and append the letters _CI to the NLS_SORT and NLS_SEARCH parameters for case-insensitive queries and sorts.
   - ❍ B. Append the letters _AI to the NLS_SORT parameter for accent-insensitive queries and sorts, and append the letters _CI to the NLS_SORT parameter for case-insensitive queries and sorts.
   - ❍ C. For accent-and case-insensitive queries and sorts, append the letters _AI_CI to the NLS_SORT parameter.
   - ❍ D. Set the NLS_SORT parameter to NO_ACCENT and/or NO_CASE.
   - ❍ E. Oracle does not support accent-and case-insensitive queries and sorts.

2. What does this SQL statement do?

SQL> SELECT last_name FROM employees
WHERE regexp_like (last_name,'^[aeiou]','i') ;

Select the best answer.

- ❍ A. Displays employee last names where the names start with any of the following letters: a, e, i, o, or u.
- ❍ B. Displays employee last names that contain any of the following letters: a, e, i, o, or u.
- ❍ C. Displays employee last names where the names start with any of the following letters: a, e, i, o, u, A, E, I, O, or U (any lower-or upper-case vowel).
- ❍ D. Displays employee last names that contain any of the following letters: a, e, i, o, u, A, E, I, O, or U.

3. Which two statements use the quote operator properly to assign a value to the PL/SQL variable?

Select the two best answers.

- ❑ A. VARIABLE := q'[Sarah's purse]'
- ❑ B. VARIABLE := q'[Sarah's purse]'q
- ❑ C. VARIABLE := bq'[Sarah's purse]'eq
- ❑ D. VARIABLE := quote'[Sarah's purse]'

Answers and Explanations

Chapter 1

1. Answer: C

Explanation A. The Database Upgrade Assistant (DBUA) is Oracle's preferred method of upgrading databases to Oracle 10g. However, it only supports directly upgrading from database releases 8.0.6, 8.1.7, 9.0.1, and 9.2.0. Therefore, you cannot use it to directly upgrade an 8.0.5 database to Oracle 10g.

Explanation B. Oracle only provides scripts for directly upgrading from releases 8.0.6, 8.1.7, 9.0.1, and 9.2.0. Therefore, you cannot use scripts to directly upgrade an 8.0.5 database to Oracle 10g.

Explanation C. For any release other than 8.0.6, 8.1.7, 9.0.1, and 9.2.0, you must first upgrade the database to one of those four releases. Thus you would first upgrade your
8.0.5 database to release 8.0.6. Now that the database is at one of the levels supported for direct upgrade, you can use the Database Upgrade Assistant (DBUA) to directly upgrade that database to 10g.

Explanation D. No, you can upgrade an 8.0.5 database to Oracle 10g; you just cannot directly upgrade it -in one step -to 10g. You will first upgrade to release 8.0.6, then directly upgrade that 8.0.6 database to Oracle 10g.

Explanation E. Oracle does not support using any "unload/load" utilities to upgrade data into an Oracle 10g database. You may have been confusing this answer with using the Export/Import utilities to upgrade data into an Oracle 10g database, which is a valid method of upgrading to 10g. Note that the Export/Import utilities tend to work best when upgrading smaller to medium-sized databases to 10g.

2. Answer: B

Explanation A. No, step (1) cannot go first because script utlu101s.sql verifies the status of the upgrade after you have upgraded the database. Also, remember that step (3) means to run the pre-upgrade script -utlu101i.sql. Knowing what these two scripts do will help you get the sequencing correct.

Explanation B. This is correct. First, run the pre-upgrade script -utlu101i.sql. Fix any discrepancies it notes and shutdown the database. STARTUP UPGRADE the database, create 10g's required SYSAUX tablespace, and run the upgrade script for your release. After the upgrade, verify it by running script utlu101s.sql. Shutdown the database and finish by recompiling any invalid objects with script utlrp.sql.

Explanation C. This answer is incorrect because the upgrade script of step (6) runs after the post-upgrade verification script of step (1) and the final re-compilation of any invalid objects in step (2). Step (6) must precede steps (1) and (2).

Explanation D. This answer is incorrect because you always run the script utlu101i.sql first. It helps you determine the pre-upgrade tasks to be completed prior to the upgrade. You must complete these tasks prior to going to STARTUP UPGRADE.

Explanation E. Starting a database in STARTUP UPGRADE mode does not automatically run any upgrade steps or scripts for you. You must manually run the scripts in steps 1, 2, and 3 yourself. Therefore, the correct sequence of steps is 3, 4, 5, 6, 1, 2.

3. Answers: B, D

Explanation A. This is not the correct answer. The minimum COMPATIBLE parameter value for 10g is 9.2.0.

Explanation B. This is correct. 9.2.0 is a valid COMPATIBLE value when upgrading to 10g. Oracle 10g requires a COMPATIBLE parameter of either 9.2.0 or 10.x.x.

Explanation C. The minimum COMPATIBLE parameter value for 10g is 9.2.0, so answering that any Oracle 8i release value would suffice is incorrect.

Explanation D. The default value of COMPATIBLE parameter in 10g is 10.x.x, which works fine. Oracle 10g requires a COMPATIBLE parameter of either 9.2.0 or 10.x.x.

Explanation E. This is incorrect because Oracle 10g requires a COMPATIBLE parameter of either 9.2.0 or 10.x.x. Ignoring the COMPATIBLE parameter's value and not inspecting it and, if necessary, setting it correctly is not an option.

Explanation F. This is incorrect because you cannot set the COMPATIBLE parameter back down to any lesser value once you have set it to 10.x.x. The only supported downgrade path is for those users who have kept COMPATIBLE=9.2.0 and have an installed 9i R2 executable.

Chapter 2

1. Answer: B

Explanation A. This is incorrect. CONTENT specifies whether the output includes data only, metadata only, or both the data and its metadata. Its values may be DATA_ONLY, METADATA_ONLY, or ALL. ALL is the default. BOTH is not an allowable value.

Explanation B. This is correct. Since CONTENT defaults to ALL, it does not need to be coded. This gives you both the data and metadata (DDL). SCHEMAS=SCOTT is optional if user id SCOTT runs the export; otherwise, it is required to dump the proper schema objects. The EXCLUDE keyword excludes any tables beginning with the letters TEMP.

Explanation C. This is incorrect because you do not use the INCLUDE parameter to exclude tables from the export. Use the EXCLUDE parameter to exclude objects from the output.

Explanation D. This is incorrect because you do not use the INCLUDE parameter to exclude tables from the export. Use the EXCLUDE parameter to exclude objects from the output.

2. Answer: B

Explanation A. This is not the correct sequence. This sequence makes sense except for the last two steps -5 then 7. Since step (7) imports the tablespaces and plugs their metadata into the target server, it must be performed before step (5), which makes the tablespaces read-write in the target database.

Explanation B. This is the correct sequence. You have some flexibility on the ordering of the first three steps (2, 4, 1), but these must occur prior to exporting the tablespace metadata (step 3). You also have some flexibility on the positioning of step 6 (when you convert the endian formats). But of the four answers listed for this question, this is the only correct sequence for these steps.

Explanation C. This sequence is incorrect because it converts the endian formats of the data (step 6) before investigating what the platforms involved in this conversion are (step 2). You need to get the information resulting from step (2) before you can correctly implement step (6).

Explanation D. This sequence is incorrect because step (1) occurs last. This step changes the tablespaces to be transported in the source database to read-only mode. You want to do this before exporting them (step 3), not afterwards. Further, step 5 converts the tablespaces in the target database to read-write mode. This should be the last step in the sequence, not the third step.

Chapter 3

1. Answer: A

Explanation A. This is the correct answer. Setting SGA_TARGET to a non-zero value and STATISTICS_LEVEL to either TYPICAL or ALL enables Automatic Shared Memory Management (ASMM). This automatically distributes memory to the Shared Pool, Large Pool, Java Pool, and standard Database Buffer Cache. ASMM does not manage the memory areas indicated by any of the other answers.

Explanation B. This answer is incorrect because ASMM does not manage memory for any database buffers defined by DB_nK_CACHE_SIZE. The only database buffer it manages memory for is the standard buffer defined by (DB_CACHE_SIZE).

Explanation C. This answer is incorrect because ASMM does not manage memory for the Log Buffer, the Streams Pool, nor Database Buffer Caches defined by parameter DB_nK_CACHE_SIZE.

Explanation D. This answer is incorrect because ASMM does not manage memory for the Log Buffer, all Database Buffer Caches, Streams Pool, nor the Fixed-SGA area and internal allocation. These are all manually configured even when ASMM is active, so you code to allocate them yourself.

Explanation E. This answer is incorrect because ASMM does not distribute memory among all of Oracle's memory areas -just among a select group of them.

Explanation F. This answer is incorrect because there is no such parameter as ASMM=Y. To enable Automatic Shared Memory Management, you set SGA_TARGET to a non-zero value and STATISTICS_LEVEL to either TYPICAL or ALL. There are no other parameters you need to set to enable ASMM.

2. Answers: A, C, F

Explanation A. This is the correct answer. When you use DBUA to create a new 10g database or update an existing database to release 10g, the new job GATHER_STATS_JOB is both created and scheduled for automatic statistics collection.

Explanation B. This is incorrect because when you use DBUA to create a new 10g database or update an existing database to release 10g, the new job GATHER_STATS_JOB is both created and scheduled for automatic statistics collection.

Explanation C. This answer is correct because when you use DBUA to either create a new 10g database or update an existing database to release 10g, the new job GATHER_STATS_JOB is both created and scheduled for automatic statistics collection.

Explanation D. This answer is incorrect because when you use DBUA to either create a new 10g database or update an existing database to release 10g, the new job GATHER_STATS_JOB is both created and scheduled for automatic statistics collection.

Explanation E. This answer is incorrect because using manual means to upgrade a database to 10g creates, but does not schedule, the statistics collection job. This answer incorrectly states that the job is scheduled.

Explanation F. This answer is correct because using manual means to upgrade a database to 10g creates the job for statistics collection, but does not schedule it.

3. Answer: A

Explanation A. This sequence is correct. Shutdown and backup the source database, then edit the SPFILE to alter, as required for ASM. Next, run an RMAN script to move database objects into an ASM disk group. Verify the new ASM database. Then at the end of the process, you can delete or archive the original database files.

Explanation B. This sequence is incorrect because it starts out by deleting the original source database files (step 1). You should not delete them until after you have successfully copied them over to ASM and verified that the new database works.

Explanation C. This sequence is incorrect because it tries to perform a cold backup of the source database (step 4) before shutting down that database (step 2). You have to shutdown the source database before you can make a cold backup of it.

Explanation D. This sequence is incorrect because it deletes or archives the source database files prior to ensuring the new ASM database works. In other words, step (6) should be preceded by step (1). Even though you have backed up the source database files in step (4), you ideally should not delete the source database files until the end of the process.

Chapter 4

1. Answers: C, E

Explanation A. This answer is incorrect because it is a valid option for the 10g cost-based optimizer.

Explanation B. This answer is incorrect because it is a valid option for the 10g cost-based optimizer.

Explanation C. This answer is correct because while 10g still includes the rule-based optimizer (RBO), the RBO is no longer supported. This means you should no longer code the two options that could result in the use of the RBO (RULE and CHOOSE).

Explanation D. This answer is incorrect because this is a valid option for the 10g cost-based optimizer.

Explanation E. This answer is correct because while 10g still includes the rule based optimizer (RBO), the RBO is no longer supported. This means you should no longer code the two options that could result in the use of the RBO (RULE and CHOOSE).

2. Answer: D

Explanation A. This answer is incorrect because 10g now allows you to collect statistics on data dictionary tables. DBMS_STATS.GATHER_DICTIONARY_STATS package can do this.

Explanation B. This answer is incorrect because 10g now allows you to collect statistics on fixed tables. DBMS_STATS.GATHER_FIXED_OBJECTS_STATS package can do this. Remember that "fixed tables" are data structures that Oracle maintains in its memory areas.

Explanation C. This answer is incorrect because you have always been able to collect statistics on user tables, even prior to Oracle 10g.

Explanation D. This answer is correct because you can use any of several packages to gather statistics on data dictionary, fixed, and user tables. Procedure DBMS_STATS.GATHER_DATABASE_STATS has been upgraded in 10g to support this.

3. Answers: A, C

Explanation A. This answer is correct because DEFAULT is in fact the correct default value for SQLTUNE_CATEGORY. Similarly, when a SQL profile is saved, the default category assigned to the profile is DEFAULT.

Explanation B. This answer is incorrect because SQLTUNE_CATEGORY is new in 10g. It defines the category to use when applying a SQL profile to a SQL statement. There is no parameter called SQLTUNE_COLLECTIONS.

Explanation C. This is correct. You assign categories to SQL profiles as they are created. When sessions connect to the database, the SQL profiles with category names that match the value of SQLTUNE_CATEGORY automatically apply to the sessions.

Explanation D. This is incorrect because SQLTUNE_CATEGORY is not part of the Database Resource Manager. It specifies the category to use when applying a SQL profile to a SQL statement.

Explanation E. This is incorrect because you can have identical SQL statements in multiple profiles as long as each profile is in a different category.

Chapter 5

1. Answers: C, D

Explanation A. This answer is incorrect because SWITCH_TIME_IN_CALL does not obsolete or replace SWITCH_TIME. Both parameters are current in Oracle 10g.

Explanation B. This answer is incorrect. For a given resource group, you can code either SWITCH_TIME_IN_CALL or SWITCH_TIME, but not both.

Explanation C. This answer is correct because for a given resource group, you can code either SWITCH_TIME_IN_CALL or SWITCH_TIME, but not both.

Explanation D. This answer is correct because the purpose of SWITCH_TIME_IN_CALL is so that a session can eventually be switched back to its original resource group, whereas SWITCH_TIME does not allow this. The additional flexibility of SWITCH_TIME_IN_CALL is why Oracle added it as a new feature in 10g.

Chapter 6

1. Answers: A, E

Explanation A. This is the correct answer. It is true that DBMS_SCHEDULER has several components that interact for flexible job scheduling. These include Programs, Schedules, Jobs, Job Classes, Windows, and Window Groups.

Explanation B. This answer is incorrect because DBMS_SCHEDULER uses Job Classes to prioritize jobs. This is one of the several components it introduces for additional scheduling flexibility, as opposed to the older DBMS_JOB package.

Explanation C. This answer is incorrect. The DBMS_JOB package is still available in Oracle 10g. However, the new package DBMS_SCHEDULER is much more powerful and flexible, and it is therefore recommended by Oracle Corp. that you use it instead of DBMS_JOB whenever possible.

Explanation D. This answer is incorrect because the new DBMS_SCHEDULER package can execute three kinds of jobs: stored programs, anonymous blocks, and operating system executables. This is specified using the JOB_TYPE parameter when creating the Job.

Explanation E. This answer is correct. DBMS_SCHEDULER uses advanced "calendaring expressions" to flexibly specify times and intervals for Jobs and Schedules.

Chapter 7

1. Answer: C

Explanation A. This answer is true, and therefore incorrect. Segment shrink frees up space below the High Water Mark (HWM), and then moves the HWM down.

Explanation B. This answer is true, and therefore incorrect. Segment shrink enables full table scans to run faster. The full scans run faster because any unused space intermingled with the table data has been removed. Data is now in fewer, contiguous blocks for faster scans.

Explanation C. This is the correct answer because it is not true. The segment shrink feature does not fix chained rows. You will still have chained rows after running segment shrink.

Explanation D. This answer is true, and therefore incorrect. Segment shrink frees up space below the High Water Mark (HWM), and then moves the HWM down.

Explanation E. This answer is true, and therefore incorrect. Segment shrink eliminates internal unused space, and results in contiguous data blocks.

2. Answer: E

Explanation A. This answer is true, and therefore incorrect. You can store one or more tables within a sorted hash cluster. Remember that "sorted hash clusters" are just a special case of "hash clusters," and hash clusters of all kinds can store one or more tables.

Explanation B. This answer is true, and therefore incorrect. Rows within a given cluster key value are sorted by sort key. This is one of the characteristics that distinguish "sorted hash clusters" from regular "hash clusters."

Explanation C. This answer is true, and therefore incorrect. Cluster keys are hashed for "sorted hash clusters," just as they are for regular "hash clusters." Remember that although sorted hash clusters are new in Oracle 10g, regular hash clusters existed in prior releases.

Explanation D. This answer is true, and therefore incorrect. You do not need to code an ORDER BY clause to retrieve data in sorted key order from sorted hash clusters. This sort order distinguishes "sorted hash clusters" from regular "hash clusters."

Explanation E. This is the correct answer because it is not true. You can create indexes on sorted hash clusters, just as you can on heap tables.

Chapter 8

1. Answers: A, E

Explanation A. This code is correct. It creates a temporary tablespace and assigns it to a temporary tablespace group. If the temporary tablespace group does not already exist, it creates that as well.

Explanation B. This code is incorrect. It is the same statement as the first answer but with the keywords "add to" included, which are incorrect and cause a syntax error.

Explanation C. This code is incorrect because it does not provide a valid tablespace group name (the '' is not a valid tablespace group name). The code runs without error and creates a temporary tablespace but does not assign it to a temporary tablespace group.

Explanation D. This code is incorrect because there is no CREATE TEMPORARY TABLESPACE GROUP command. You create a temporary tablespace group indirectly by creating or altering a temporary tablespace to be part of a new temporary tablespace group.

Explanation E. This code is correct. It assigns an existing temporary tablespace to the temporary tablespace group named in the statement, and will create the temporary tablespace group if it does not already exist.

Chapter 9

1. Answers: A, F

Explanation A. This is true. You enable fast incremental backups by creating a change block tracking file of at least 10 megabytes in size.

Explanation B. This is not true. While fast incremental backups do use a change block tracking file, the new background process that supports this feature is called CTWR (Change Tracking Writer). There is no such background process as FIBW (Fast Incremental Backup Writer).

Explanation C. This is not true. You must specifically enable the feature by issuing the SQL statement: ALTER DATABASE ENABLE BLOCK CHANGE TRACKING. This statement either names a file to use for change block tracking, or omits that parameter if you are using Oracle Managed Files (OMF).

Explanation D. This is not true. The SQL statements used to turn this feature on and off are dynamic, assuming you use an SPFILE. The statements to use are ALTER DATABASE ENABLE BLOCK CHANGE TRACKING and ALTER DATABASE DISABLE BLOCK CHANGE TRACKING.

Explanation E. This is not true. Oracle implements fast incremental backups through its new block change tracking file and the new background process CTWR. The new block change tracking file allows Oracle to identify only the data blocks that have been changed, thereby speeding performance by avoiding the need to read the entire file (both changed and unchanged data blocks).

Explanation F. This is true. Fast incremental backups use these two components to allow Oracle to only process changed blocks (ignoring unchanged blocks), thereby allowing for much faster incremental backups.

Chapter 10

1. Answer: B

Explanation A. This answer is incorrect because, while it lists the two parameters required, their sequence is reversed. Always define DB_RECOVERY_FILE_DEST_SIZE first, then DB_RECOVERY_FILE_DEST.

Explanation B. This answer is correct because it lists the two parameters in the correct order. You define the size of the flash recovery area first, then its location.

Explanation C. This answer is incorrect because the parameters must include the word DEST.

Explanation D. This answer is incorrect because the parameters must include the word DEST.

2. Answers: B, C

Explanation A. This is not true. The flashback database feature allows you to "flash back" the entire database to a specified prior point in time. You cannot select and flash back any certain individual tablespaces.

Explanation B. This is true. View V$DATABASE has a new column FLASHBACK_ON you can inspect to see if the flashback database feature is enabled.

Explanation C. This is true. RVWR is the new background process that the flashback database feature relies on to keep track of data changes. RVWR writes changed data from the flashback buffer in the SGA to the flashback logs. The flashback logs reside in the flash recovery area.

Explanation D. This is not true. Yes, you set parameter DB_FLASHBACK_RETENTION_TARGET to tell Oracle how far into the past you want the flashback logs to go. However, its value is set in minutes -not seconds.

Explanation E. This is not true. Flashback database uses the flashback logs to go back to the specified point in time. It does not use the Undo data for this purpose. This is the reason why Oracle keeps the flashback logs -to go back to some point in time. Oracle may then apply Redo logs to go forward to a consistent transactional state.

Chapter 11

1. Answer: B

Explanation A. This answer is incorrect. All parameters are correct, except for the option of ONLINE or OFFLINE status. The SYSAUX tablespace is required to be ONLINE for the database to be up and active.

Explanation B. This answer is correct. All the listed parameters are required for the SYSAUX tablespace.

Explanation C. This answer is incorrect because the SYSAUX tablespace is required to be PERMANENT. Therefore, it is incorrect to state that it is TEMPORARY.

Explanation D. This answer is incorrect because the SYSAUX tablespace is required to be READ WRITE. Therefore, it is incorrect to state that it is READ ONLY.

Explanation E. This answer is incorrect because the SYSAUX tablespace is required to be PERMANENT and READ WRITE. Therefore, it is incorrect to state that it is TEMPORARY and READ ONLY.

Explanation F. This is incorrect because the SYSAUX tablespace requires SEGMENT SPACE MANAGEMENT AUTO. Recall that if you do not specify the SEGMENT SPACE MANAGEMENT clause, in all Oracle releases including 10g, it defaults to SEGMENT SPACE MANAGEMENT MANUAL.

2. Answer: B

Explanation A. This answer is incorrect. There is no new procedure called DBMS_SYSAUX.MOVE_PROCEDURE. You need to query table V$SYSAUX_OCCUPANTS to find out what the specific "move procedure" is to run to relocate any particular SYSAUX occupant.

Explanation B. This answer is correct. The table V$SYSAUX_OCCUPANTS gives you the "move procedure" you would run to relocate any particular SYSAUX occupant. Remember that not all occupants can be moved. Those that cannot be moved will show blanks in the retrieved MOVE_PROCEDURE column.

Explanation C. This answer is incorrect. The new COPY_FILE procedure is used to read a local file and create a copy of it on the local system. It is not used to relocate occupants from the SYSAUX tablespace.

Explanation D. This answer is incorrect. You can relocate occupants out of the SYSAUX tablespace. The correct answer explains the proper procedure. It is true that the SYSAUX tablespace must be ONLINE and available for Oracle to function, but this is not affected by using the proper procedure to move one of its relocatable occupants to another tablespace.

Chapter 12

1. Answer: D

Explanation A. This is incorrect because the policy does not audit "all SQL statements." As indicated by the parameter STATEMENT_TYPES, it only audits UPDATE, INSERT, and DELETE statements (not SELECT statements).

Explanation B. This is incorrect because the policy does not audit "all SQL statements." As indicated by the parameter STATEMENT_TYPES, it only audits UPDATE, INSERT, and DELETE statements (not SELECT statements). Furthermore, AUDIT_COLUMNS_OPTS specifies ALL_COLUMNS, so both the FIRST_NAME and the LAST_NAME must be touched to trigger an audit record.

Explanation C. This is incorrect because the parameter AUDIT_COLUMNS_OPTS specifies ALL_COLUMNS, so both the FIRST_NAME and the LAST_NAME must be touched to trigger the audit record.

Explanation D. This is correct. Parameter STATEMENT_TYPES lists the kinds of SQL statements required to trigger an audit record, while AUDIT_COLUMN_OPTS says that ALL_COLUMNS must be involved to trigger an audit record (the alternative is ANY_COLUMNS, which means accessing any one of the listed columns triggers an audit record.)

Chapter 13

1. Answer: B

Explanation A. This is incorrect because there is no NLS_SEARCH parameter. You only need to set appropriate values for NLS_SORT to alter queries and sorts. Also, you do not need to set values for other valid NLS parameters, such as NLS_LANGUAGE or NLS_COMP.

Explanation B. This is correct. You only need to set new values for the NLS_SORT parameter to affect queries and sorts. Do this by appending the letters _AI to the NLS_SORT parameter for accent-insensitive queries and sorts, or append the letters _CI to the NLS_SORT parameter for case-insensitive queries and sorts.

Explanation C. This is incorrect. You either append the letters _AI or _CI to the NLS_SORT parameter, you do not append both.

Explanation D. This is incorrect, as there are no such settings as NO_ACCENT and NO_CASE.

Explanation E. This is incorrect. This is a new feature for 10g.

2. Answer: C

Explanation A. This is incorrect because the "i" specifies a case-insensitive comparison. We are thus searching for last names that begin with either an upper-case or lower-case vowel.

Explanation B. This is incorrect because the regular expression contained in the brackets [and] searches for a single character. The ^ notation states that this will be the first character in the line (not anywhere in the name). Also, the "i" indicates a case-insensitive search, so the upper-case vowels will also match the search pattern.

Explanation C. This is correct. This is a regular expression comparison using the new 10g function REGEXP_LIKE. The ^ letter represents the beginning of a line, so we are inspecting the first character in the line. The brackets [and] enclose a list of alternative letters that occur after the line beginning. The letter "i" says to make a case-insensitive comparison. So we retrieve names beginning with upper-or lower-case vowels.

Explanation D. This is incorrect because the regular expression contained in the brackets [and] searches for a single character. The ^ notation states that this will be the first character in the line (not anywhere in the name).

3. Answer: A

Explanation A. This code correctly uses the quote operator. The quote operator allows you to choose your own quote mark delimiter and eliminates the need for additional quotation strings in character literals. The quotation mark delimiter is defined using the quote operator q followed by a single quote. The delimiters used here are [and] .

Explanation B. This is incorrect because you only need the first q as the quote operator.

Explanation C. This is incorrect because the quote operator is a q (not a bq followed by an eq).

Explanation D. This is incorrect because the quote operator is a q, not the word quote.