6 Space and Storage Management Enhancements

CERTIFICATION OBJECTIVES

In Chapter 4, you learned about the new automatic database alerts. In this chapter, you'll see how you can reduce space-related errors by proactively managing tablespace usage. Undo space management is tricky sometimes, especially when you run long transactions. You can use tablespace usage alerts to manage normal tablespaces as well as the undo tablespace.

A major problem for many Oracle DBAs is how to reclaim unused space in the database. You need to run SQL scripts to find out the amount of free space, and possibly move or re-create database objects to recover the excess space. Due to the time these database-reorganizing efforts consume, quite often, the free space is never reclaimed. In Oracle Database 10*g*, you can shrink segments in line and in place. Thus, you can reclaim unused space without affecting database availability.

In this chapter, you'll learn about the new Segment Advisor, which helps you isolate candidates for shrinking and to track segment growth over time, so you can plan your space requirements correctly. You'll next learn how to use the Undo Advisor and the Redo Logfile Size Advisor. This chapter also introduces you to the new data structure called the sorted hash cluster, which helps you store data sorted by nonprimary key columns.

You read about the new SYSAUX tablespace in Chapter 1. In this chapter, you'll get the details of creating and managing the SYSAUX tablespace. You'll learn about temporary tablespaces and assigning temporary tablespace groups to users. You'll also look at topics like renaming tablespaces and creating a default temporary tablespace. Large databases need large datafiles. The new bigfile tablespaces address this need by enabling the creation of large tablespaces with just a single data file. This chapter concludes by showing you how to copy files using the database server.

Let's start with a quick look at the new concept of proactive tablespace management in Oracle Database 10*g*.

CERTIFICATION OBJECTIVE 6.01

Proactive Tablespace Management

In Oracle Database 10*g*, by default, all tablespaces have built-in alerts that notify you when the free space in the tablespace goes below a certain predetermined threshold level. There are two default alert thresholds: *critical* and *warning*. The new background process, MMON, is responsible for monitoring tablespace free space and sending out alerts. Oracle issues an alert each time the database crosses either one (warning or critical) of the alert thresholds. When you clear the proactive alert by assigning exam watch

If you are migrating to Oracle Database 10g, Oracle turns off the automatic tablespace alerting mechanism by default. You can use the `DBMS_SERVER_ALERT` package to set the alert thresholds.

more space to the tablespace, Oracle issues another alert to acknowledge the fixing of the problem.

By default, Oracle sends out a warning alert when your tablespace is 85 percent full and a critical alert when the tablespace is 97 percent full. You may change these default settings, or even turn the entire alerting mechanism off, if you so desire. The `DBA_THRESHOLDS` view will provide information about the thresholds in your database.

The proactive tablespace alerts come with some caveats:

- You can't set alerts for dictionary-managed tablespaces. You can set alerts only for locally managed tablespaces.
- When you take a tablespace offline or make it read-only, you must turn the alerting mechanism off (using the `DBMS_SERVER_ALERT` package). You can turn the alerts back on after you change the status of the tablespaces again.
- For the undo tablespace, the active and unexpired extents together constitute the used space in that tablespace. The same default warning and critical thresholds (97 percent and 85 percent, respectively) apply to the undo tablespace as well.

You can manage the tablespace usage alerts by setting or modifying the alert thresholds. You can do this with the OEM Database Control or by using the PL/SQL package `DBMS_SERVER_ALERT`. Let's first learn how to manage the tablespace space usage alert using Database Control.

exam watch

If you get a snapshot too old alert warning, you may need to increase the size of your undo tablespace. In addition, you may consider increasing the length of the undo retention period. Note that you'll get a maximum of only one undo alert during any 24-hour period.

Using the Database Control to Manage Thresholds

You can use the OEM Database Control to set database-wide default thresholds. Here is a summary of the steps you need to follow to use the Database Control for this purpose:

1. From the Database Control home page, click the Manage Metrics link.
2. In the Manage Metrics page, click the Edit Thresholds button

3. From the Edit Thresholds page, you can set thresholds for the Tablespace Usage metric. Click the Specify Multiple Thresholds button. This will bring you to the Specify Multiple Thresholds: Tablespace Used page. You can set thresholds for a specific tablespace from here.

Using the DBMS_SERVER_ALERT Package

You can also use the `DBMS_SERVER_ALERT` package to manage database thresholds. In Chapter 4, you were introduced to two of the package's procedures: `SET_THRESHOLD` and `GET_THRESHOLD`. Using these procedures, you can perform the following tasks:

- Set warning and critical thresholds
- Set database-wide default values for tablespace space usage
- Find out the levels of the current default tablespace usage thresholds
- Turn off the space-usage tracking mechanism for any tablespace

CERTIFICATION OBJECTIVE 6.02

Reclaiming Unused Space

A segment's high-water mark (HWM) shows the highest point of space usage ever reached by that segment. If you have used 80 percent of a table segment's space by inserting rows into that segment, for example, the HWM for that segment will be at 80 percent. Later on, even if you delete half the rows, the table's HWM remains at 80 percent. This has a detrimental effect on full-table scans and index scans, as Oracle will scan the table all the way to the HWM, even if currently there is very little data in the table due to deletions over time.

A table segment with large number of deletions will thus be sparsely populated, with the deletions leading to fragmentation of the space, leaving several gaps below its HWM level. Oracle will move the HWM lower only when you truncate all the table rows. The only way for you to reclaim the space you allotted to a table is to drop the table altogether or truncate it.

on the job

In segments using Automatic Segment Storage Management (ASSM), you don't really have an HWM. Instead, there is the high HWM, above which no blocks are formatted, and the low HWM, below which all blocks are formatted.

exam watch

The segment-shrinking capability is termed an on line and in place operation. It's on line because users can continue to access the tables during the shrinking operation. The operation is in place because you don't need any duplicate or temporary database space during the segment-shrinking operations.

In previous versions of Oracle, the only way to compact the unused pockets of space in table or index segments was to move the object or redefine it. These reorganizations sometimes could be very time-consuming.

In Oracle Database 10*g*, you can now use the new *segment-shrinking capability* to make sparsely populated segments give their space back to their parent tablespace. You can reduce the HWM, thus compacting the data within the segments. In Oracle Database 10*g*, you can shrink the following types of segments:

- Tables, including index-organized tables (IOTs)
- Partitions and subpartitions of a table
- Indexes
- Materialized views and materialized view logs

Oracle handles the shrinking operation internally as an insert/delete operation. Note that any triggers on the tables will not fire when you perform the insertions and deletions, since you are only moving the data around, not changing it. Each batch of insertions/deletions constitutes one transaction. When you shrink a table to compact space, the indexes on the table remain intact and are in a usable state after the shrinking operation.

Restrictions on Shrinking Segments

The following are some restrictions that apply to shrinking segments:

- You can't shrink any segment that uses free lists. This means that you can only shrink segments that use Automatic Segment Space Management.
- You must enable row movement (for heap-organized segments). Enabling row movement lets you specify whether Oracle can move a table row to a different location. Oracle may need to move a row, for example, during table compression or an update operation on partitioned data. During a

exam watch

The following objects can't be shrunk: clustered tables, tables with LONG columns, tables with on-commit materialized views, tables with ROWID-based materialized views, LOB segments, IOT mapping and overflow segments, and tables with function-based indexes.

segment-shrinking operation, ROWIDs may change, thus necessitating the enabling of row movement.

- Even with ASSM-based segments, you can't shrink tables that are part of a cluster, tables with LONG columns, certain types of materialized views, and certain types of IOTs. You also can't shrink tables with function-based indexes.

Manual Segment Shrinking

You can use simple SQL commands to shrink segments. However, there is a prerequisite to running the command. You must first enable row movement for any segment that you want to shrink. You can enable row movement by using the `ENABLE ROW MOVEMENT` clause of the `ALTER TABLE` command, as shown here:

```
SQL> alter table test ENABLE ROW MOVEMENT;
```

Of course, if you've already specified the `ENABLE ROW MOVEMENT` clause at table creation time, you won't need to issue any commands to enable row movement before starting the segment-shrinking operation.

exam watch

You must enable row movement before shrinking segments because the segment-shrinking operation may cause the ROWIDs in heap-organized segments to change.

on the job

By default, row movement is disabled at the segment level.

The segment-shrinking operation compacts fragmented space in the segments and *optionally* frees the space.

There are two phases in a segment-shrinking operation:

- **Compaction phase** During the compaction phase, the rows in a table are compacted and moved toward the left side of the segment. You thus make the segment dense, but the HWM remains where it was. The recovered space is not yet released to the tablespace. During this phase, you can issue DML statements and queries on a segment while it is being shrunk. Oracle holds locks on packets of the rows involved in the DML operations. If you have any long-running queries, Oracle can read from all the blocks that have technically been reclaimed during the shrinking operation. Of course, this capability is dependent on the time interval you specified for your undo retention parameter.

- **Adjustment of the HWM/releasing space phase** During the second phase, Oracle lowers the HWM and releases the recovered free space under the old HWM to the parent tablespace. Oracle locks the object in an exclusive mode. This implies that you can't issue any `INSERT, UPDATE`, and `DELETE` DML statements against the segment. During this phase, Oracle invalidates any cursors referencing the segment as well. This second phase lasts for a very short period.

on the job

During the compacting phase, the object is online and available, but during the second phase, the object becomes unavailable, albeit very briefly.

The basic statement to shrink segments performs both phases of the segment-shrinking operation (first compacting, then resetting the HWM and releasing the space) in sequence. Here's the statement:

```
SQL> alter table test SHRINK SPACE;
```

Once you issue the `ALTER TABLE` *`table`* `SHRINK SPACE` command, Oracle will first compact the segment, and then reset the HWM level and yield the freed space to the tablespace.

on the job

To avoid disruption of DML activity and cursor invalidation problems, perform just the compact operations during peak levels of activity.

Since the second phase, the resetting of the HWM, will affect DML operations, you may not want to use it when a large number of users are connected to the database. Instead, you may want to issue the following command, which compacts only the space in the segment:

```
SQL> alter table test SHRINK SPACE COMPACT;
```

This way, during peak hours, the database will merely compact the space in the segment. During off-peak hours, you can issue the `ALTER TABLE test SHRINK SPACE` command, and this will finish the shrinking process by performing the second phase.

exam watch

If you use the `ALTER TABLE table_name SHRINK SPACE` command to shrink a table, Oracle will compact the space, adjust the HWM, and release the space. If you add the `COMPACT` clause, Oracle will only compact the space in the table segment.

If you use the `CASCADE` option during a segment-shrinking operation, all the dependent segments will be shrunk as well. For example, if you shrink a table, all the dependent index segments will be automatically shrunk. Here's how you specify the `CASCADE` option:

```
SQL> alter table test SHRINK SPACE CASCADE;
```

Shrinking Segments Using the Database Control

You can easily perform all segment-shrinking operations using the OEM Database Control interface. First, make sure that you have row movement enabled for the segment you wish to shrink. You can enable row movement using the Database Control, by following the links Schema, Tables, Edit Tables, then Options. Here, you can enable row movement by choosing that option for your segment.

You can shrink a table segment by first choosing the Tables link under the Schema listing. On the Tables page, select Shrink Segments from the Actions field and click Go. This takes you to the Shrink Segment page, shown in Figure 6-1. On this page, you can choose to shrink a segment in the two ways described in the previous section:

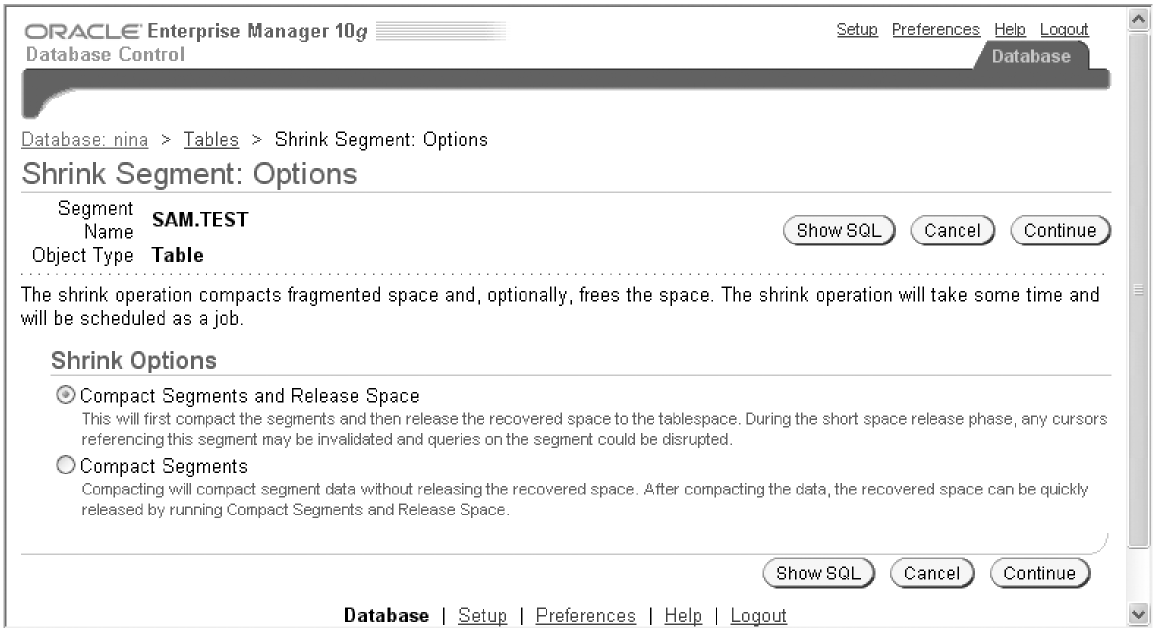
- Compact segments and release space
- Compact segments

Using the Segment Advisor

How do you know which of your segments is a good candidate for shrinking? How do you find out which objects have unused space that they won't be needing in the future?

Using the new Segment Advisor, you can easily identify the segments that are good candidates for shrinking. The Segment Advisor bases its recommendations on the amount of fragmentation within an object. It determines whether objects have enough space to be reclaimed, taking into account the future space requirements. It bases its estimates of the future space requirements of an object on historical trends. Besides helping you select candidates for shrinking, the Segment Advisor is also helpful in sizing new database objects. The following sections describe how to use the advisor for both purposes.

FIGURE 6-1 The Database Control Shrink Segment page


ORACLE Enterprise Manager 10g
Database Control
Setup Preferences Help Logout
Database
Database: nina > Tables > Shrink Segment: Options

Shrink Segment: Options

Segment Name SAM.TEST
Object Type Table

The shrink operation compacts fragmented space and, optionally, frees the space. The shrink operation will take some time and will be scheduled as a job.

Shrink Options

[✓ Compact Segments and Release Space]
This will first compact the segments and then release the recovered space to the tablespace. During the short space release phase, any cursors referencing this segment may be invalidated and queries on the segment could be disrupted.

Compact Segments
Compacting will compact segment data without releasing the recovered space. After compacting the data, the recovered space can be quickly released by running Compact Segments and Release Space.

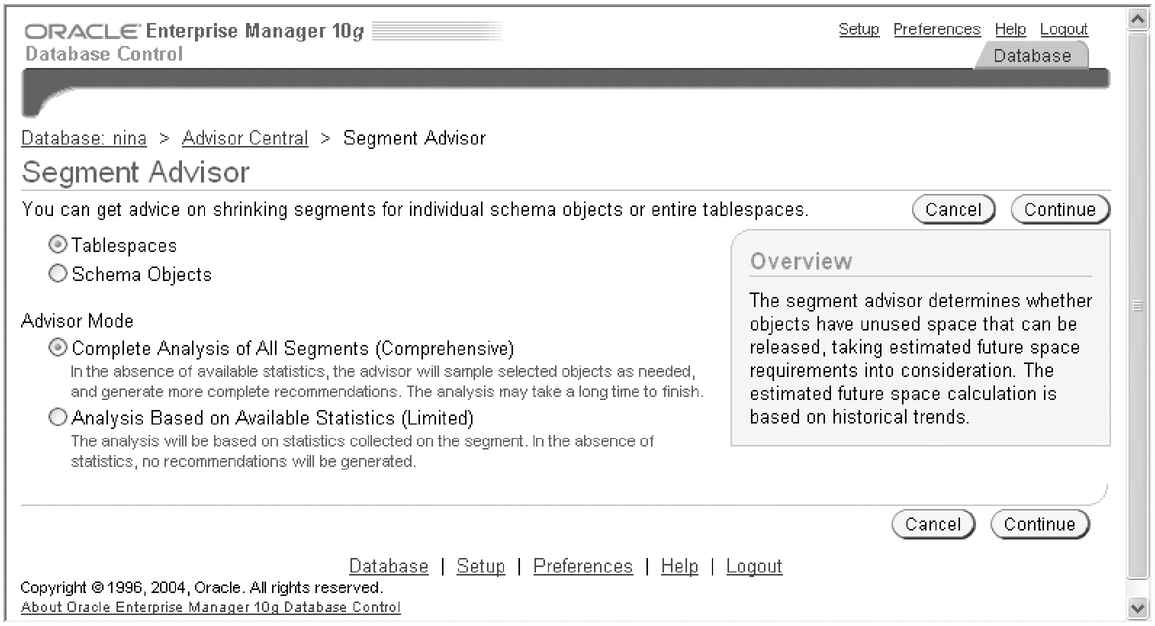
Choosing Candidate Objects for Shrinking

You can invoke the Segment Advisor at either the individual segment level or tablespace level. You can call the Segment Advisor from the Database Control's Advisor Central page (from the Database Home page, click Advisor Central in the Related Links section, and then click Segment Advisor), Tablespaces page, or Schema Object page. Figure 6-2 shows the main Segment Advisor page.

You can run the Segment Advisor in two modes:

- **Comprehensive analysis** The Segment Advisor will perform an analysis regardless of whether there are prior statistics. If there aren't any prior statistics, the Segment Advisor will sample the objects before generating its recommendations. This analysis is more time-consuming.
- **Limited analysis** This analysis is based strictly on the statistics collected on the segment. If there aren't any statistics or an object, the advisor won't perform any analysis.

FIGURE 6-2 The Database Control Segment Advisor page



Database: nina > Advisor Central > Segment Advisor

Segment Advisor

You can get advice on shrinking segments for individual schema objects or entire tablespaces.

[✓ Tablespaces]
Schema Objects

Advisor Mode

[✓ Complete Analysis of All Segments (Comprehensive)]
In the absence of available statistics, the advisor will sample selected objects as needed, and generate more complete recommendations. The analysis may take a long time to finish.

Analysis Based on Available Statistics (Limited)
The analysis will be based on statistics collected on the segment. In the absence of statistics, no recommendations will be generated.

Overview

The segment advisor determines whether objects have unused space that can be released, taking estimated future space requirements into consideration. The estimated future space calculation is based on historical trends.

Copyright © 1996, 2004, Oracle. All rights reserved.
About Oracle Enterprise Manager 10g Database Control

Where does the Segment Advisor get the data for its recommendations? The Automatic Workload Repository (AWR) collects all space-usage statistics during its regular snapshot collection. The Segment Advisor, to estimate future segment space needs, uses the *growth trend report* based on the AWR space-usage data. The growth trend report predicts future segment growth patterns as well as traces past growth trends. The Segment Advisor stores its results in the AWR as well.

on the job

You must use locally managed tablespaces if you want the database to produce growth trend reports.

Estimating Object Size

When you create a new database, it is customary to use spreadsheets to help you figure out how much space to allocate to new database objects like tables and indexes. You can now use the Segment Advisor to determine your future segment resource usage. You provide the structure of your table (column data types, column sizes, and the PCTFREE parameter's size) or index and the number of rows, and the advisor will tell you how much space you need.

In order to use the segment resource estimation feature, follow these steps:

1. From the Database Control home page, click the Administration tab.
2. Under the Storage section, click the Tables link.
3. Click the Create button to create a new table.
4. You'll now be on the Create Table page. Under the Columns section, specify your column data types. Then click the Estimate Table Size button.
5. On the Estimate Table Size page, specify the estimated number of rows in the new table, under Projected Row Count. Then click the Estimated Table Size button. This will show you the estimated table size.

CERTIFICATION OBJECTIVE 6.03

Using the Undo and Redo Logfile Size Advisors

The Segment Advisor is not the only space-related advisor in Oracle Database 10*g*. You can also use the Undo Advisor and the Redo Logfile Size Advisor to help you manage the undo and redo activity in your database. Let's look at how to use these advisors.

The Undo Advisor

The Undo Advisor helps you perform the following tasks:

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

You can access the Undo Advisor through the Database Control, as follows:

1. From the Database Control home page, click Administration.
2. Click the Undo Management button, which is under the Instance list.
3. From the main Undo Management page, click the Undo Advisor button in the right corner.

The AWR collects statistics on undo activity in the database. In Oracle Database 10g, the database uses this AWR information to decide on the appropriate number of undo segments to bring online when you start the instance. Oracle uses the same AWR data to decide on the number of undo segments to bring online when you switch undo tablespaces. In both cases, the ramp-up time for the database to bring undo segments online is much lower than in previous versions.

The Redo Logfile Size Advisor

You can control database recovery time by setting the appropriate size for the mean time to recover (MTTR) from a crash. The `FAST_START_MTTR_TARGET` initialization parameter determines instance recovery time after a crash.

In order to be of the optimal size, your redo log files should be set just large enough that the database isn't performing more checkpoints than that required by the value of your `FAST_START_MTTR_TARGET` parameter. Small log file sizes may mean that the database writer will be performing incremental checkpointing more frequently than what is dictated by your MTTR value. As you know, frequent log switching will tend to drive the incremental checkpoint activity, causing the database writer to perform excessive disk I/O. Ideally, this activity should be driven by the MTTR target rather than by log switching. So, too small a size for the redo log files could result in excessive checkpointing, which in turn, increases disk activity, which is detrimental to system performance. However, if you set the redo log file size too large, there isn't enough checkpointing, which makes it harder to meet your MTTR requirements.

on the job

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

The Redo Logfile Size Advisor will make recommendations about the smallest online redo log files you can use. This advisor recommends that you size your redo log files to at least the recommended minimum size. A new column in the `V$INSTANCE_RECOVERY` view, `OPTIMAL_LOGFILE_SIZE`, shows you the optimal size of the redo log file for your `FAST_START_MTTR_TARGET` setting.

You can access the Redo Logfile Size Advisor through the Database Control, as follows:

1. From the Database Control home page, click Administration.
2. Under the Storage section, choose Redo Log Groups.

3. Select any redo log group, and then choose the Sizing Advice option from the Action drop-down list.
4. Click Go to get the redo log file size advice.

CERTIFICATION OBJECTIVE 6.04

Tablespace Enhancements

Oracle Database 10*g* includes several interesting enhancements pertaining to tablespaces. These enhancements involve the management the SYSAUX tablespace, default permanent tablespaces, temporary tablespace groups, and bigfile tablespaces. Let's start our discussion with the management of the new SYSAUX tablespace.

Managing the SYSAUX Tablespace

Oracle Database 10*g* mandates the creation of the SYSAUX tablespace, which serves as an auxiliary to the SYSTEM tablespace.

Until now, the SYSTEM tablespace was the default location for storing objects belonging to components like the Workspace Manager, Logical Standby, Oracle Spatial, Logminer, and so on. The more features the database offered, the greater was the demand for space in the SYSTEM tablespace. In addition, several features had to be accommodated in their own repositories, like the Enterprise Manager and its Enterprise Manager repository, and the Oracle Text feature with its DRSYS location. On top of all this, you had to create a special tablespace for the STATSPACK repository.

To alleviate this pressure on the SYSTEM tablespace and to consolidate all the repositories for the various Oracle features, Oracle Database 10*g* offers the SYSAUX tablespace as a centralized, single storage location for various database components. Using the SYSAUX tablespace offers the following benefits:

- You have fewer tablespaces to manage, because you don't need to create a separate tablespace for many database components. You just assign the SYSAUX tablespace as the default location for all the components.
- There is reduced pressure on the SYSTEM tablespace, which is used as the default tablespace for several features.
- If you are using Real Application Clusters (RAC) with raw devices, you'll have fewer raw devices to manage, since every tablespace under RAC requires at least one raw device.

The size of the SYSAUX tablespace depends on the size of the database components that you'll store in it. Therefore, you should base your SYSAUX tablespace sizing on the components and features that your database will use. Oracle recommends that you create the SYSAUX tablespace with a minimum size of 240MB.

Creating the SYSAUX Tablespace

If you use the Oracle Database Configuration Assistant (DBCA), you can automatically create the SYSAUX tablespace when you create a new database, whether it is based on the seed database or is a completely new, built from scratch, user-defined database. During the course of creating a database, the DBCA asks you to select the file location for the SYSAUX tablespace. When you upgrade a database to Oracle Database 10g, the Database Upgrade Assistant (DBUA) will similarly prompt you for the file information for creating the new SYSAUX tablespace.

exam watch

The SYSAUX tablespace is mandatory, whether you create a new tablespace or migrate to Oracle Database 10g.

You can create the SYSAUX tablespace manually during database creation time. Here's the syntax for creating the SYSAUX tablespace (showing only the relevant parts of the `CREATE DATABASE` statement):

```
Create database mydb
user sys identified by abc1def
user system identified by  uvw2xyz
...
sysaux datafile  '/u01/oracle/oradata/mydb/sysaux01.dbf' size 500M reuse
...
;
```

You can set only the datafile location when you create the SYSAUX tablespace during database creation, as shown in the preceding example. Oracle sets all the

exam watch

What happens if you omit the SYSAUX creation clause from the `CREATE DATABASE` statement? Oracle will create the SYSAUX tablespace anyway. Oracle will create both the SYSTEM and SYSAUX tablespaces automatically, with their datafiles in system-determined default locations. If you are using Oracle Managed Files (OMF), the datafile location will be dependent on the OMF initialization parameters. If you include the `DATAFILE` clause for the SYSTEM tablespace, you must use the `DATAFILE` clause for the SYSAUX tablespace as well, unless you are using OMF.

other attributes, which are mandatory. You cannot later change the location or other attributes with the ALTER TABLESPACE command.

Oracle requires that the SYSAUX tablespace have the following attributes:

- Permanent
- Read/write
- Locally managed
- Automatic segment space management

Altering the SYSAUX Tablespace

If you have the SYSDBA system privilege, you can alter the SYSAUX tablespace, using the same ALTER TABLESPACE command that you use for your other tablespaces. Here's an example:

```
SQL> alter tablespace sysaux add datafile
'/u01/app/oracle/prod1/oradata/sysaux02.dbf' size 500M;
```

Usage Restrictions for the SYSAUX Tablespace

Although the use of the ALTER TABLESPACE command may make it seem like the SYSAUX tablespace is similar to the other tablespaces in your database, several usage features set the SYSAUX tablespace apart. Here are the restrictions:

- You can't drop the SYSAUX tablespace by using the DROP TABLESPACE command during normal database operation.
- You can't rename the SYSAUX tablespace during normal database operation.
- You can't transport a SYSAUX tablespace.

Relocating SYSAUX Occupants

As you recall, the purpose behind the creation of the SYSAUX tablespace is the need for storing data belonging to the large number of database components in Oracle Database 10*g*. What happens if you later decide that you want to move components out of the SYSAUX tablespace to a different tablespace? You may want to do this in response to a severe space pressure on the SYSAUX tablespace, for example.

You can monitor space usage of the SYSAUX tablespace occupants by using the new data dictionary view V$SYSAUX_OCCUPANTS. Here's the structure of this view:

```
SQL> desc v$sysaux_occupants
 Name                                      Null?    Type
 ----------------------------------------- -------- --------------
 OCCUPANT_NAME                                      VARCHAR2(64)
 OCCUPANT_DESC                                      VARCHAR2(64)
 SCHEMA_NAME                                                 VARCHAR2(64)
 MOVE_PROCEDURE                                              VARCHAR2(64)
 MOVE_PROCEDURE_DESC                                         VARCHAR2(64)
 SPACE_USAGE_KBYTES                                          NUMBER
```

The SPACE_USAGE_KBYTES column will tell you how much of the SYSAUX tablespace's space each of its occupants is currently using. The MOVE_PROCEDURE column tells you the specific procedure you must use in order to move a given occupant out of the SYSAUX tablespace.

exam watch

You can move occupants into and out of the SYSAUX tablespace. In order to perform the move operation, you must have the SYSDBA privilege and ensure that tablespace is online. For each user of the SYSAUX tablespace, there is a specific move procedure to effect the move.

Let's say that you decide to move an occupant of the SYSAUX tablespace into a new tablespace. You must first find out which procedure you must use to move this occupant, since each of the occupants of the SYSAUX tablespace may need a separate procedure to allow this.

Here's a simple query that demonstrates how to find out which procedure to employ for moving an occupant of the SYSAUX tablespace:

```
SQL>  select occupant_name, move_procedure
  2*  from v$sysaux_occupants
OCCUPANT_NAME                                                     MOVE_PROCEDURE
---------------------------------------------------------------- ---------------
LOGMNR                                          SYS.DBMS_LOGMNR_D.SET_TABLESPACE
LOGSTDBY                                       SYS.DBMS_LOGSTD BY.SET_TABLESPACE
STREAMS
AO                                                          DBMS_AW.MOVE_AWMETA
XSOQHIST                                              DBMS_XSOQ.Olapi MoveProc
SM/AWR
SM/ADVISOR
SM/OPTSTAT
SM/OTHER
STATSPACK
ODM                                                                   MOVE_ODM
SDO                                                             MDSYS.MOVE_SDO
WM                                                           DBMS_WM.move_proc
ORDIM
ORDIM/PLUGINS
ORDIM/SQLMM
EM                                            emd_maintenance.move_em_tblspc
TEXT                                                           DRI_MOVE_CTXSYS
ULTRASEARCH                                                            MOVE_WK
JOB_SCHEDULER
20 rows selected.
SQL>
```

Once you find out the name of the correct procedure to employ in order to move a SYSAUX occupant, you can perform the move itself, using the appropriate procedure. For example, you can move the occupant named WM in the following manner:

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

This preceding `MOVE` statement will move the user WM from the SYSAUX tablespace to the DRSYS tablespace.

exam watch

You can't relocate the following occcupants of the SYSAUX tablespace: STREAMS, STATSPACK, JOB_ SCHEDULER, ORDIM, ORDIM/PLUGINS, ORDIM/SQLMM, and SMC.

Renaming Tablespaces

In previous versions of Oracle, a major limitation concerning tablespaces was that you couldn't rename a tablespace once you created it. In Oracle Database 10g, you can rename tablespaces easily by using a simple command (from users to users_new in this example):

```
SQL> alter tablespace users rename TO users_new;
```

The ability to rename tablespaces saves substantial amounts of administrative time and effort. You can rename permanent as well as temporary tablespaces.

You can now transport a tablespace even if the target database contains identically named tablespaces as the source. You just need to rename the identically named tablespaces in the target database before you start the transport of the tablespaces.

Oracle advises you not to use tablespace names as part of the filenames, since Oracle will not rename the datafiles as part of the tablespace-renaming procedure.

If you perform recovery on a datafile whose header contains the old tablespace name, once you recover past the tablespace renaming point, the datafile header will replace the old tablespace name with its new name.

There are some restrictions and requirements, however, in renaming tablespaces:

- Your compatibility level must be set to 10.0 or higher.
- You can't rename the SYSTEM or SYSAUX tablespace, or offline tablespaces.
- Tablespace identifiers remain the same.
- If the renamed tablespace is the default tablespace for any user before the renaming, it will continue being the default after renaming.
- If the tablespace is read-only, the datafile headers aren't updated, although the control file and the data dictionary are.

For an undo tablespace, the name change has different implications, based on whether you used the init.ora file or the SPFILE to start the instance. If you are using the init.ora file, Oracle writes a message stating that you should change the value (location) of the `UNDO_TABLESPACE` parameter in the init.ora file. If you used the SPFILE to start the instance, Oracle will automatically write the new name for the undo tablespace in your SPFILE, provided you specified the tablespace name as the undo tablespace name in your SPFILE.

Default Permanent Tablespaces

Every user in an Oracle database has a default permanent tablespace, which is where the objects created by that user will be placed by default. For example, if the user doesn't specify a tablespace in a `CREATE TABLE` statement, that table will go into the user's default tablespace.

What happens when you don't specify a default tablespace for a user? Well, in previous versions of Oracle, the SYSTEM tablespace automatically became the default permanent tablespace for that user. That's not exactly a smart thing, since you don't want just any user to start creating objects in the SYSTEM tablespace.

In Oracle Database 10g, if you don't specify a default tablespace during user creation, the user isn't automatically expected to use the SYSTEM tablespace as the default tablespace. You can now create a default permanent tablespace that all users will use as the default permanent tablespace, when you don't specifically assign a permanent tablespace during user creation.

exam watch

The users SYS, SYSTEM, and OUTLN continue to use the SYSTEM tablespace as their default permanent tablespace. All the other users must use the new default permanent tablespace.

You can use the Database Control or SQL statements (`ALTER DATABASE`) to designate a default permanent tablespace. You can also specify a default permanent tablespace during database creation time. In the following sections, we'll look at the different ways to designate the default permanent tablespace.

on the job

You can't drop a default permanent tablespace without attending to some preliminaries first. You must first assign a different tablespace as the default tablespace for all the users, and only then proceed to dropping the current default tablespace.

Specifying the Default Tablespace During Database Creation

You can specify a default permanent tablespace during database creation by using the new DEFAULT TABLESPACE clause. Here's a (partial) database creation statement showing how to use the DEFAULT TABLESPACE clause to specify a default permanent tablespace named deftbs for all users in the database:

```
SQL> create database mydb
    user SYS identified by abc1def
    user SYSTEM identified by uvw2xyz
    default tablespace deftbs datafile …
    …;
```

The DEFAULT TABLESPACE clause isn't mandatory. If you don't use one, Oracle will still create the database, with the SYSTEM tablespace set as the default tablespace, as in previous versions. However, Oracle recommends that you specify the DEFAULT TABLESPACE clause explicitly, so your users aren't forced to use the SYSTEM tablespace as their default tablespace.

Using the Database Control to Specify the Default Tablespace

You can use the OEM Database Control to quickly assign and change the default permanent tablespace for your database. Here's the procedure for designating a default permanent tablspace using the Database Control:

1. From the Database Control home page, click Administration.
2. From the Storage Section, choose the Tablespaces link.
3. Click the tablespace you want to designate as the default permanent tablespace. If the tablespace doesn't exist, first create it before you proceed any further.
4. From the Edit Tablespace page, select the Set As Default Permanent Tablespace option in the Type section. Then click Apply.

Using SQL to Specify the Default Tablespace

You can designate any tablespace as the default permanent tablespace after database creation by using the ALTER TABLESPACE command, as follows:

```
SQL> alter database default tablespace new_tbsp;
```

When you designate a default tablespace in this manner, all current users will be assigned to the new default permanent tablespace. New users are automatically allotted this tablespace as their default permanent tablespace as well.

Viewing Default Tablespace Information

To find out what the name of your current default permanent tablespace is, simple execute the following query, which uses the view `DATABASE_PROPERTIES`:

```
SQL> select property_value from database_properties
  2  where property_name='DEFAULT_PERMANENT_TABLESPACE';
PROPERTY_VALUE
---------------------------
USERS
SQL>
```

Temporary Tablespace Groups

Large transactions may sometimes run out of temporary space. Large sort jobs, especially on tables with many partitions, strain temporary tablespaces. Oracle Database 10*g* introduces the concept of a *temporary tablespace group*, which actually represents a list of temporary tablespaces. Using a tablespace group, a user can utilize multiple temporary tablespaces simultaneously in different sessions.

Using temporary tablespace groups, rather than the usual single temporary tablespace, provides several benefits, including the following:

- SQL queries are less likely to run out of sort space, because the query can now use several temporary tablespaces for sorting.
- You can now specify multiple default temporary tablespaces at the database level.
- Parallel execution servers in a parallel operation will efficiently utilize multiple temporary tablespaces.
- A single user can simultaneously use multiple temporary tablespaces in different sessions.

How does one go about creating a temporary tablespace group? Well, interestingly, you don't explicitly create a temporary tablespace group. When you assign the first temporary tablespace to a tablespace group, you automatically create the temporary tablespace group.

Characteristics of Temporary Tablespace Groups

Here are some of the main characteristics of a temporary tablespace group:

- There is a minimum of one tablespace in a temporary tablespace group. There is no explict maximum number of tablespaces.
- If you delete all members from a temporary tablespace group, the group is automatically deleted as well.
- A temporary tablespace group has the same namespace as tablespaces.
- You can't name a temporary tablespace the same as the name of any tablespace group.
- When you need to assign a temporary tablespace to a user, you can just use the temporary tablespace *group* name, instead of the actual temporary tablespace. This principle also applies when you create the default temporary tablespace for the database.

exam watch

Just as a temporary tablespace group is created when you assign the first temporary tablespace to it, the group is also deleted when you remove the last temporary tablespace from the group.

Creating a Temporary Tablespace Group

You implicitly create a temporary tablespace group when you specify the TABLESPACE GROUP clause in a CREATE TABLESPACE statement, as shown here:

```
SQL> create temporary tablespace temp_old  tempfile
'/u01/oracle/oradata/temp01.dbf'
     size 500M
     tablespace group group1;
```

The preceding SQL statement will create a new temporary tablespace, temp_now, along with the new tablespace *group* named group1. Oracle creates the new tablespace group since the temporary tablespace temp_old is the first tablespace assigned to it.

You can also create a temporary tablespace group by specifying the same TABLESPACE GROUP clause in an ALTER TABLESPACE command, as shown here:

```
SQL> alter tablespace temp_old
  2  tablespace group group1;
Tablespace altered.
SQL>
```

In the preceding statement, since there wasn't a prior temporary tablespace group named group1, Oracle will create a new group with that name.

Creating Temporary Tablespaces Not Belonging to a Group

To tell Oracle *not* to allocate a temporary tablespace to a tablespace group, specify a pair of quotation marks ('')—for the tablespace group name. Here's an example:

```
SQL> create temporary tablespace temp_new tempfile 'temp_new1.f'
     SIZE 500m TABLESPACE GROUP '';
```

The preceding statement creates a temporary tablespace called temp_new, which is like the traditional temporary tablespaces and doesn't belong to a temporary tablespace group.

on the job

Use the `TEMPFILE` clause, not the `DATAFILE` clause, when you allocate space to a temporary tablespace.

If you completely omit the `TABLESPACE GROUP` clause, you'll create just a regular temporary tablespace, which is not part of any temporary tablespace group, as shown in the following example.

```
SQL> create temporary tablespace temp_new2
  2  tempfile 'c:\oracle\tmp3.f' size 5M;
  Tablespace created.
SQL>
```

exam watch

If you specify the NULL tablespace group (for example, `CREATE TEMPORARY TABLESPACE GROUP … GROUP ''`), it is equivalent to the normal temporary tablespace creation statement (without any groups).

Adding a Tablespace to Temporary Tablespace Group

You add a temporary tablespace to group by using the `ALTER TABLESPACE` statement. You can also change the group a temporary tablespace belongs to in the same way. For example, you can specify that your new tablespace temp_new belong to the group named group1 by issuing the following command:

```
SQL> alter tablespace temp_new tablespace group   group1;
```

Note that the database would create a new group with the name group1 if that group did not already exist.

Setting a Group As the Default Temporary Tablespace for the Database

You can use a temporary tablespace group as your mandatory default temporary tablespace for the database. If you issue the following command, all users without a default tablespace can use both the temp_old and temp_new tablespaces (assuming that they are part of the group group1) as their default temporary tablespaces.

```
SQL> alter database default temporary tablespace group1;
Database altered.
SQL>
```

This assigns all the tablespaces in group1 as the default temporary tablespaces for the database.

Using a Temporary Tablespace Group When You Create and Alter Users

When you create new users, you can now assign users to a temporary tablespace group, instead of the usual single temporary tablespace. Here's an example:

```
SQL> create user sam identified by sam
  2  DEFAULT TABLESPACE users
  3* TEMPORARY TABLESPACE group1;
User created.
SQL>
```

Once you create a user, you can also use the `ALTER USER` statement to change the temporary tablespace group of the user. Here's the SQL statement to do this:

```
SQL> alter user sam temporary tablespace group2;
```

Viewing Temporary Tablespace Group Information

You can use the new data dictionary view `DBA_TABLESPACE_GROUPS` to manage the temporary tablespace groups in your database. Here is a simpley query on the view that shows the names of all tablespace groups:

```
SQL> select group_name, tablespace_name
  2  from dba_tablespace_groups;
GROUP_NAME         TABLESPACE_NAME
----------------------------------
GROUP1                  TEMP01
SQL>
```

You can also use the DBA_USERS view to find out which temporary tablespaces *or* temporary tablespace groups are assigned to each user. Here's an example:

```
SQL> select username, temporary_tablespace from dba_users;
USERNAME                        TEMPORARY_TABLESPACE
------------------------------ ------------------------------
SYS                             TEMP
SYSTEM                          TEMP
SAM                             GROUP1
SCOTT                           TEMP
. . .
SQL>
```

Bigfile Tablespaces

Oracle Database 10*g* can contain up to 8 exabytes (8 million terabytes) of data. Don't panic, however, thinking how many milions of datafiles you need to manage in order to hold this much data. You now have the option of creating really big tablespaces, called, appropriately, *bigfile tablespaces*. A bigfile tablespace (BFT) contains only *one very large file*. Depending on the block size, a bigfile tablespace can be as large as 128 terabytes.

In previous versions, you always had to juggle with the distinction between datafiles and tablespaces. Now, using the bigfile concept, Oracle has made a tablespace logically equal to a datafile, by creating the new one-to-one relationship between tablespaces and datafiles. With Oracle Managed Files, datafiles are completely transparent to you when you use a bigfile tablespace, and you can directly deal with the tablespace itself in many kinds of operations.

exam watch

The traditional tablespaces are now referred to as smallfile tablespaces. Smallfile tablespaces are the default tablespaces in Oracle Database 10g. You can have both smallfile and bigfile tablespaces in the same database.

Here's a summary of the many benefits offered by using bigfile tablespaces:

- You need to create only as many datafiles as there are tablespaces. Fewer datafiles mean that you can use a smaller setting for the DB_FILES initialization parameter. (The DB_FILES parameter determines the maximum number of datafiles that can be open.)
- You don't need to constantly add datafiles to your tablespaces.
- You have simplified datafile management in large databases, because you deal with tablespaces directly, not many datafiles.

- The database has a significant increase in storage capacity, because you don't reach the maximum files limitation quickly when you use bigfile tablespaces.
- You can set the `CREATE DATABASE` clause `MAXDATAFILES` to a lower size, and thus reduce the size of the control file. The `MAXDATAFILES` parameter specifies the control file's initial space allocation to datafile information.

Restrictions on Using Big File Tablespaces

There are few restrictions on using bigfile tablespaces. You can use bigfile tablespaces only if you use a locally managed tablespace with ASSM. Oracle also recommends that you use bigfile tablespaces along with a Logical Volume Manager (LVM) or the Automatic Storage Management (ASM) feature, which support striping and mirroring. Otherwise, you can't really support the massive datafiles that underlie the bigfile tablespace concept. Both parallel query execution and RMAN backup parallelization would be adversely impacted if you used bigfile tablespaces without striping.

To avoid creating millions of extents when you use a bigfile tablespace in a very large (greater than a terabyte) database, Oracle recommends that you change the default extent allocation policy (`AUTOALLOCATE`) to `UNIFORM` and set a very high extent size. In databases that aren't very large, Oracle recommends that you stick to the default `AUTOALLOCATE` policy and simply let Oracle take care of the extent sizing.

on the job

Locally managed undo and temporary tablespaces can be bigfile tablespaces, even if you don't use ASSM.

Making Bigfile the Default Tablespace Type

You can now specify bigfile as the default tablespace type during database creation. If you don't explicitly specify bigfile as your default tablespace type, your database will have smallfile tablespaces as the default. Smallfile tablespaces are nothing but the normal traditional tablespaces that you use currently in Oracle databases. Here's a portion of the `CREATE DATABASE` statement showing how you specify a bigfile tablespace as the default:

```
create database test
set default bigfile tablespace
 …
```

You can also dynamically change the default tablespace type to bigfile (or smallfile), thus making all tablespaces you subsequently create that type. Here's an example that shows how to set the default tablespace type in your database to bigfile from now on:

```
SQL> alter database set default bigfile tablespace;
```

Once you set the default type of your tablespace, all the tablespaces you subsequently create will be of the bigfile type, unless you manually override the default setting, as shown in the next section.

Creating a Bigfile Tablespace Explicitly

Irrespective of which default type you choose—bigfile or smallfile—you can always create a bigfile tablespace by specifying the type explicitly in the `CREATE TABLESPACE` statement, as shown here:

```
create bigfile tablespace bigtbs
datafile '/u01/oracle/data/bigtbs_01.dbf' size 100G
...
```

In the preceding statement, the explicit specification of the `BIGFILE` clause will overrirde the default tablespace type, if it was a smallfile type. Note that if your default tablespace type was bigfile, then you can use the keyword `SMALLFILE` to override the default type when you create a tablespace.

When you specify the `CREATE BIGFILE TABLESPACE` *`tablespace_name`* clause, Oracle will automatically create a locally managed tablespace with ASSM. You can specify the datafile size in kilobytes, megabytes, gigabytes, or terabytes.

on the job

Creating bigfile tablespaces on operating system platforms that don't support large files will limit tablespace capacity to the maximum file size that the operating system can support.

Migrating Database Objects

You can migrate database objects from a smallfile tablespace to a bigfile tablespace or vice versa, if you are planning to change your tablespace type. You can migrate the objects using the `ALTER TABLE … MOVE` or the `CREATE TABLE AS SELECT` command. Alternatively, you may use the Data Pump export and import tools to move the objects between the two types of tablespaces, as described in Chapter 2.

Altering a Bigfile Tablespace's Size

You can use the RESIZE and AUTOEXTEND clauses in the ALTER TABLESPACE statement to modify the size of a bigfile tablespace.

The RESIZE clause lets you resize a bigfile tablespace directly, without using the DATAFILE clause, as shown here:

```
SQL> alter tablespace bigtbs resize 120G;
```

The AUTOEXTEND clause enables automatic file extension, again without referring to the datafile. Here's an example:

```
SQL> alter tablespace bigtbs autoextend on next  20G;
```

Note that both these space extension clauses can be used directly at the tablespace level, not the file level. Thus, both of these clauses provide datafile transparency—you deal directly with the tablespaces and ignore the underlying datafiles.

Viewing Bigfile Tablespace Information

You can gather information about the bigfile tablespaces in your database by using the following data dictionary views:

- DBA_TABLESPACES
- USER_TABLESPACES
- V$TABLESPACE

All three views have the new column BIGFILE, whose value indicates whether a tablespace is of the bigfile type (YES) or smallfile type (NO).

You can also use the following query to determine the default tablespace type for your database:

```
SQL> select property_value
  2  from database_properties
  3* where property_name='DEFAULT_TBS_TYPE';
PROPERTY_VALUE
--------------
SMALLFILE
SQL>
```

The following query helps in finding out which of your tablespaces have bigfile tablespaces:

```
SQL> select tablespace_name, bigfile FROM dba_tablespaces;
TABLESPACE_NAME        BIG
-----------------    --------
SYSTEM                 NO
RBS                    NO
USERS                  NO
TEMP                   NO
TOOLS                  NO
INDX                   NO
DRSYS                  NO
SYSAUX                 NO
BIG_TBS                YES
9 rows selected.
SQL>
```

Bigfile Tablespaces and ROWID Formats

Bigfile tablespaces use the extended ROWID format, and the only supported way to extract the ROWID components is by using the DBMS_ROWID package. The DBMS_ROWID package lets you create and extract the components of ROWIDs. Smallfile and bigfile tablespaces have different formats; therefore, you must specify the tablespace type when you use the DBMS_ROWID package. You can specify the tablespace type by using the new parameter TS_TYPE_IN, which can take the values BIGFILE and SMALLFILE.

For bigfile tablespaces, there is only a single file, with the relative file number always set to 1024. The encoded block number consists of a concatenation of the datafile number and the data block that contains the row. For smallfile tablespaces, the ROWIDs are in the format *Object# - File# - Block# - Row#*. For bigfile tablespaces, the format is *Object# - Block# - Row#*.

The encoded block number for bigfile tablespaces can be much larger than those for traditional smallfile tablespaces. Bigfile tablespace block numbers are relative to the tablespace the rows belong to, and are unique as well.

on the job

If you are using bigfile tablespaces to construct an internal or external ROWID string, use the ROWID_CREATE function of the DBMS_ROWID package. The relative file number (RELATIVE_FNO) will always be 1024 for bigfile tablespaces.

You can use the DBMS_ROWID package to get the ROWID formats of a bigfile tablespace, as shown here:

```
SQL> select distinct DBMS_ROWID.ROWID_RELATIVE_FNO
(ROWID,'BIGFILE ')
FROM test_rowid;
```

Since the ROWID formats are different for bigfile and smallfile tablespaces, the database needs to know the tablespace type when it is creating and extracting ROWIDs. Several parameters belonging to the package DBMS_ROWID need to use the new input parameter TS_TYPE_IN, which informs Oracle whether the tablespace type is bigfile or smallfile. The TS_TYPE_IN parameter can take the values BIGFILE or SMALLFILE. It is a part of the DBMS_ROWID procedures ROWID_INFO, ROWID_BLOCK_NUMBER, and ROWID_RELATIVE_FNO.

CERTIFICATION OBJECTIVE 6.05

Using Sorted Hash Clusters

Suppose you have a query that requires that the rows be returned in a specific order. You can't specify that Oracle save the table data in the same order as the query might require it to be output. Oracle bases its row placement in the table purely on storage conditions. To guarantee a specific row order, you must use an ORDER BY clause. The ORDER BY clause will return the rows in a guaranteed order, but often involves a substantial use of memory and CPU time. Oracle Database 10*g* introduces the new *sorted hash cluster* storage scheme, which can help you retrieve sorted data faster and with much less resource usage.

What Is a Sorted Hash Cluster?

When you are dealing with a set of tables that share common columns and are frequently used together in queries, it is better to create a table cluster. A *table cluster* is a group of tables that Oracle physically stores together in the same data blocks.

For example, suppose that your employees and departments table share the department_id column. When you cluster the employees and departments tables, Oracle stores all rows for each department from both the employees and departments tables in the same physical location (same data blocks). A *cluster key value* is the value of the cluster key columns for a particular row. In our example, department_id is your cluster key. Since you store related rows physically together, your disk I/O will be less and disk-access times will be faster. You might gain substantially by clustering tables that you frequently join in many of your queries.

A *hash cluster* groups data according to a hash function it applies to the cluster key value of each row. Oracle then stores all rows with identical cluster key values together on disk. Hash clusters are recommended when you frequently use equality queries (for example, return all rows for dept_id=20). Oracle hashes the cluster key value, and the hashed key value will point directly to the area on disk where the rows are stored.

exam watch

The selection of the order key column for a sorted hash cluster is important, since you would need additional sorting when you use an ORDER BY clause on a suffix of the sort key columns. The same is true when you use nonsorted key columns.

In a *sorted hash cluster*, rows corresponding to individual values of the hash function are sorted on specific columns. The rows are organized as lists of sorted rows, with each list corresponding to a separate value of the hash key column. Each list will then have its rows sorted in the order specified by the sort key columns. In other words, the table's rows are already presorted by the sort key column.

Here are some of the main features of sorted hash clusters:

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

Defining Sort Hash Clusters

In order to define a sorted hash cluster, you must first create the cluster itself, and then create the actual table or tables that are part of the cluster. Let's look at both of these steps.

exam watch

For a given cluster key value, the cluster can return the rows in ascending or descending order. If you want to retrieve rows in ascending order, the `ORDER BY` clause isn't mandatory.

Creating the Cluster

When you create a regular hash cluster, you need to specify only the cluster key. To create a sorted hash cluster, you must specify an additional sort key as well. Thus, the sort hash cluster has two keys: the *cluster key* and the *sort key*. Here's the statement that creates a cluster called message_cluster:

```
SQL> create cluster call_cluster
  2  (call_number NUMBER
  3  , call_timestamp NUMBER SORT
  4  , call_duration NUMBER SORT)
  5  hashkey 10000
  6  single table hash is order_number
  7  size  50;
  Cluster created.
SQL>
```

In the call_cluster cluster, the cluster key is call_number, and the combination of call_time and call_duration is the sort key. `SIZE` indicates the space in bytes that is necessary to store the cluster key metadata.

Creating Cluster Tables

In order to define a table as part of the sorted hash cluster, you must use the `CLUSTER` clause when you create the table. You must also ensure that the table's cluster key columns and sort key columns are specified in the same order as in the parent cluster. Here's a simple example, showing the creation of the calls table, based on the call_ cluster cluster created in the previous section.

```
SQL> create table calls
  2  (call_number NUMBER
  3  , call_timestamp NUMBER
  4  , call_duration  NUMBER
  5  , call_info      VARCHAR2(50))
  6  CLUSTER call_cluster
  7  (origin_number,call_timestamp,call_duration);
Table created.
SQL>
```

The calls table is linked to the call_cluster cluster with the keyword `CLUSTER`. This table stores the call records for a company, with the origin_number column representing the originating telephone number. The calls from each originating number are processed during the preparation of customer bills. The idea is that by storing them in a sorted hash cluster, it is easy to process the bills on a first-in, first-out basis, if they are already sorted on the sort key, consisting of the call_timestamp and call_duration columns.

In our example, if you wanted to query the orders table for a certain order_number (the cluster key), the cluster key value is hashed to its metadata entry in the sorted hash cluster segment. This provides the sorted list of rows for that particular hash key column.

CERTIFICATION OBJECTIVE 6.06

Copying Files Using the Database Server

You normally use UNIX or Windows utilities to copy or move binary files. Oracle Database 10*g* introduces a new way to copy files using the database server itself. By using this new file-copy method, you bypass the operating system altogether. The `DBMS_FILE_TRANSFER` package helps you copy binary files to a different location on the same server or transfer files between Oracle databases.

on the job

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

File Copy Requirements

There are a few prerequisites that you must satisfy in order to use the `DBMS_FILE_TRANSFER` package to copy files locally or remotely:

- The maximum file size is 2 terabytes, and the file must be in multiples of 512 bytes.
- You can't perform any character set conversion during the copy process.
- All nonprivileged users of the database must have explicit privileges before they can access a file created by using `DBMS_FILE_TRANSFER`.

Let's see how you can use the DBMS_FILE_TRANSFER package to perform various types of file-copy procedures.

Copying Files on a Local System

In order to copy files on a local file system, you must use the COPY_FILE procedure of the DBMS_FILE_TRANSFER package. In the following example, a file named exm_old.txt is being copied from the /u01/App/Oracle directory to the /u01/App/Oracle/Example directory on the same server. In the process of copying the file, its name is changed to exm_new.txt.

First, you must first create the necessary directories: one for the source and the other for the destination directory. You can use the following commands to create the directories:

```
create directory source_dir as '/u01/app/oracle';
create directory dest_dir as  '/u01/app/oracle/example';
```

Once you create the directories, you must run the DBMS_TRANSFER.COPY_FILE procedure to copy the file from the source to the destination directory. Here's the code:

```
begin
  dbms_file_transfer.copy_file(
  source_directory_object       =>  'SOURCE_DIR',
  source_file_name              =>  'exm_old.txt',
  destination_directory_object  =>  'DEST_DIR',
  destination_file_name         =>  'exm_new.txt');
end;
```

on the job

A user other than the SYSTEM user must have the READ privilege on the source directory and the WRITE privilege on the destination directory in order to execute the DBMS_TRANSFER.COPY_FILE procedure.

Transferring a File to a Different Database

In the previous section, you saw how you can copy files on a local system. You can also transfer files to make copies of local files on a remote file system. You use the PUT_FILE procedure of the DBMS_FILE_TRANSFER package to perform this remote file transfer.

Let's use a simple example to demonstrate this interserver file-transfer capability. As when you copy files on a local system, you first need to create the source and destination directories. Then execute the PUT_FILE procedure as follows:

```
begin
  dbms_file_transfer.put_file(
  source_directory_object       =>  'SOURCE_DIR',
  source_file_name              =>  'exm_old.txt',
  destination_directory_object  =>  'DEST_DIR',
  destination_file_name         =>  'exm_new.txt'
  destination_database          =>   'US.ACME.COM');
end;
```

The preceding example uses the DESTINATION_DATABASE clause, which enables you to transfer datafiles from your local server to another database. In order to transfer a file the other way around, you must replace the PUT_FILE procedure with the GET_FILE procedure.

You can monitor the progress of all your file-copy operations using the V$SESSION_LONGOPS view. You may also use the Oracle Scheduler to schedule the file-copy operations.

CERTIFICATION SUMMARY

This chapter started with a review of proactive tablespace alerts. You then learned about the new feature that allows you to reclaim space by shrinking segments. Next, you saw how to use the Segment Advisor to estimate the size of database objects as well as perform other useful tasks. You also quickly reviewed the two space-related advisors: the Redo Logfile Size Advisor and the Undo Advisor.

The major portion of this chapter was spent discussing several interesting tablespace-related features, including the ability to rename tablespaces, the mandatory SYSAUX tablespace, the use of temporary tablespace groups, and default permanent tablespaces. You also learned how the new bigfile tablespaces can make your job a lot easier when you are dealing with very large databases.

Next, the chapter discussed sorted hash clusters, which reduce the need to sort data. Lastly, you were introduced to the new Oracle Database 10*g* feature that enables you to copy files directly through the database server.

INSIDE THE EXAM

The certification exam asks you several questions about the new tablespace enhancements in Oracle Database 10g. Expect at least one question on the new SYSAUX tablespace, probably about the process of relocating occupants in and out of that tablespace. What are the special features (restrictions) of a SYSAUX tablespace, compared with a normal tablespace (you can't rename, drop, or transport a SYSAUX tablespace).

You must understand how to use the advisor that will help you perform segment shrinking (the Segment Advisor). You must also know the prerequisites for a segment-shrinking operation. Of course, you are going to see a question or two testing your knowledge of the difference between the `SHRINK SPACE` command with and without the COMPACT option. When does Oracle compact space, and when does it adjust the high-water mark and release space held by a compacted object?

The test focuses on the new temporary tablespace group concept. The test expects you to know the properties of temporary tablespace groups. You must also know how to create temporary tablespace groups, as well as assign and deassign temporary tablespaces to temporary tablespace groups. Expect a question on the new bigfile tablespace type. Specifically, you must be aware of what the Oracle Database 10g default tablespace type is and how to change it.

You must remember that you can now create a default permanent tablespace in Oracle Database 10g. The test may examine your understanding of the `DEFAULT TABLESPACE` command to create a permanent default tablespace during database creation. How do you alter the current default permanent tablespace? What is the default permanent tablespace when you create a new database using the Database Configuration Assistant (DBCA)?

The tablespace renaming feature is new in Oracle Database 10g, and the exam may test your understanding of when you can and cannot rename tablespaces. What are the best practices for tablespace renaming?

You must also understand how to use the Redo Logfile Size Advisor to configure optimal online redo log file sizes.

✓ TWO-MINUTE DRILL

Proactive Tablespace Management

- ❑ All Oracle Database 10*g* tablespaces have built-in alerts.
- ❑ The warning tablespace alert is when the tablespace is 85 percent full, and the critical alert is at 97 percent full.
- ❑ Use the `DBMS_SERVER_ALERT` package to set alert thresholds.
- ❑ Use the `SET_THRESHOLD` and `GET_THRESHOLD` procedures to manage database thresholds.
- ❑ You can set alert thresholds for only locally managed tablespaces.

Reclaiming Unused Space

- ❑ The high-water mark (HWM) is the highest point of space usage ever reached by a segment.
- ❑ You can use the new segment-shrinking capability to make segments release space that is fragmented or empty.
- ❑ You can shrink tables, indexes, partitions, subpartitions, materialized views, and materialized view logs.
- ❑ Segment shrinking in Oracle Database 10*g* is *on line* and *in place*.
- ❑ After a table-shrinking operation, the indexes are intact and in a usable state.
- ❑ You can shrink only segments in ASSM managed tablespaces
- ❑ You must first enable row movement before you can shrink a table, because ROWIDs could change during a shrinking operation.
- ❑ By default, row movement is disabled at the segment level.
- ❑ You perform a segment-shrinking operation in two stages: compacting data and releasing the space.
- ❑ DML statements can continue during the first stage, but not during the second stage of shrinking a segment.
- ❑ Oracle recommends that you don't perform the second stage of a segment-shrinking operation during busy periods.
- ❑ Use the `CASCADE` option to shrink all the dependent objects of a segment.

❑ You can run the Segment Advisor in the comprehensive or limited mode.

❑ To estimate future space needs, the Segment Advisor uses the growth trend reports.

Using the Undo and Redo Logfile Size Advisors

❑ The Undo Advisor helps you set the undo retention period.

❑ You can also set the size of the undo tablespace with the help of the Undo Advisor.

❑ The `FAST_START_MTTR_TARGET` parameter sets the instance recovery time.

❑ Too small redo log files relative to the MTTR setting lead to excessive checkpointing.

❑ If the redo log files are too large, there isn't enough checkpointing.

❑ You have optimally sized redo logs when the database isn't performing any more checkpointing than that required by the `FAST_START_MTTR_TARGET` parameter.

Tablespace Enhancements

❑ The SYSAUX tablespace is auxiliary to the SYSTEM tablespace and holds data belonging to various Oracle components.

❑ The SYSAUX tablespace is mandatory in Oracle Database 10*g*.

❑ If you don't create a SYSAUX tablespace, Oracle will create one by default.

❑ The SYSAUX tablespace keeps the pressure low on the SYSTEM tablespace and makes it easier to manage the various database components.

❑ A SYSAUX tablespace must be read/write, permanent, locally managed, and use ASSM.

❑ You can't drop or rename the SYSAUX tablespace during normal tablespace operations.

❑ You must use a specific "move procedure" to move an occupant of the SYSAUX tablespace to a different tablespace.

❑ You can rename a tablespace by using the `ALTER TABLESPACE` command.

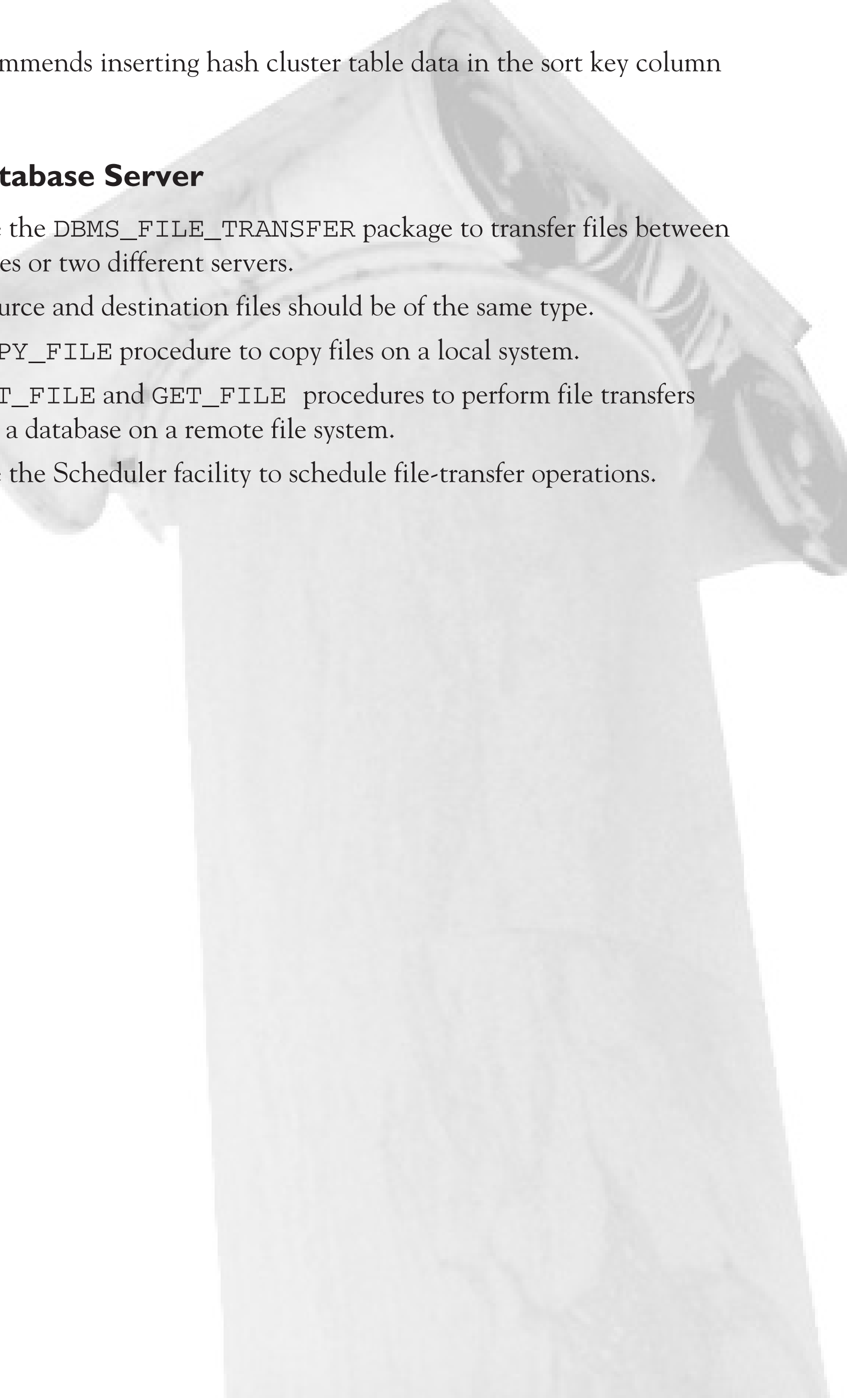
❑ You can't rename the SYSTEM, SYSAUX, and offline tablespaces.

❑ You may specify a default permanent tablespace during database creation.

- ❑ You can use multiple temporary tablespaces simultaneously in various sessions, by using the temporary tablespace group concept.
- ❑ You create a temporary tablespace group automatically when you assign a temporary tablespace to it.
- ❑ A user can use multiple temporary tablespaces in different sessions.
- ❑ A bigfile tablespace consists of only a single large file.
- ❑ In a bigfile tablespace, there is a one-to-one relationship between a datafile and a tablespace.
- ❑ If you use bigfile tablespaces, you can reduce the value of the `MAXDATAFILES` clause during database creation and use a smaller setting for the `DB_FILES` initialization parameter.
- ❑ Oracle recommends using bigfile tablespaces with file systems that support striping and mirroring.
- ❑ You can have bigfile and smallfile tablespaces together in the same database.
- ❑ The default tablespace for a database continues to be the traditional smallfile tablespace.
- ❑ You use the `DBMS_ROWID` package to extract the `ROWID` components of bigfile tablespaces.
- ❑ You must use a new parameter called `TS_TYPE_IN` in the `DBMS_ROWID` procedures. The new parameter will help you determine the tablespace type.
- ❑ The encoded block number in the `ROWIDs` can be much larger for bigfile tablespaces.

Using Sorted Hash Clusters

- ❑ Sorted hash clusters enable the retrieval of sorted data faster than using an `ORDER BY` clause.
- ❑ A hash cluster groups data according to a hash function it applies to the cluster key value of each row.
- ❑ The table rows are organized as lists of sorted rows, with each row corresponding to a separate value of the hash column.
- ❑ You would still need to use an `ORDER BY` clause on a suffix of the sort key columns.



- ❑ Before you can create the tables that are part of a cluster, you must create the cluster itself.
- ❑ If you want to retrieve data in ascending order, the `ORDER BY` clause isn't mandatory.
- ❑ Oracle recommends inserting hash cluster table data in the sort key column order.

Copying Files Using the Database Server

- ❑ You can use the `DBMS_FILE_TRANSFER` package to transfer files between two databases or two different servers.
- ❑ Both the source and destination files should be of the same type.
- ❑ Use the `COPY_FILE` procedure to copy files on a local system.
- ❑ Use the `PUT_FILE` and `GET_FILE` procedures to perform file transfers to and from a database on a remote file system.
- ❑ You can use the Scheduler facility to schedule file-transfer operations.

SELF TEST

The following questions will help you measure your understanding of the material presented in this chapter. Read all the choices carefully, because there might be more than one correct answer. Choose all correct answers for each question.

Proactive Tablespace Management

1. For which two types of tablespaces can you set proactive tablespace alerts?

   A. Locally managed
   B. Dictionary-managed
   C. Read-only
   D. Online

2. Which of the following is true about alerts?

   A. You'll get a maximum of one undo alert per every 12-hour period.
   B. You'll get a maximum of one undo alert during a 24-hour period.
   C. You'll get a maximum of one tablespace alert during a 124-hour period.
   D. You'll get a maximum of one tablespace alert during a 24-hour period.

3. Which one of the following is responsible for sending out tablespace alerts?

   A. `DBMS_SERVER_ALERT` package
   B. MMON background process
   C. Database Control
   D. Scheduler

Reclaiming Unused Space

4. Which of the following segments cannot be shrunk?

   A. Heap-organized table segments
   B. Index-organized table segments
   C. Index segments
   D. Undo segments

5. The DBA issues the following command:

```
SQL> alter table test SHRINK SPACE
```

   Which of the following would be the result of this statement?

A. There would be immediate release of the free space.
B. There would be only a compacting of the data, not a release of data.
C. There would be no impact on DML operations in the database.
D. There would be an impact on the DML operations in the database.

6. What is the first step before you start shrinking a segment?

A. Make sure there are indexes on all the table segments.
B. Enable row movement for any segment you want to shrink.
C. Disable row movement for any segment you want to shrink.
D. Issue the `ALTER TABLE … COMPACT` command.

Using the Undo and Redo Logfile Size Advisors

7. What does the Undo Advisor help you do?

A. Set the undo retention period
B. Set the undo interval
C. Set the size of the undo tablespace
D. Figure out the number of undo segments you must create

8. What does the initialization parameter `FAST_START_MTTR_TARGET` determine?

A. The instance recovery time
B. The database restore time
C. The number of redo log groups
D. The undo retention period

9. Which of the following is true of optimally sized redo log files?

A. They should perform very frequent checkpointing.
B. They should perform very infrequent checkpointing.
C. The amount of checkpointing isn't related to the size of the redo log file.
D. They should perform just the right amount of checkpointing as that required by your MTTR.

Tablespace Enhancements

10. If you don't explicitly specify the SYSAUX tablespace during database creation, what happens?

A. The database creation will fail.
B. Oracle will create a default SYSAUX tablespace.

C. Oracle will create a default SYSTEM tablespace.

D. Oracle will let you create a SYSAUX tablespace later on.

11. Which one of the following is true when you rename a default permanent tablespace for a user?

A. You'll need to create a new default tablespace for the user before you can do anything.

B. The tablespace will continue to be the default tablespace after you rename it.

C. You must first remove all the current users from the tablespace.

D. You can't rename a default permanent tablespace.

12. Which of the following statements is true when you are considering dropping the current default permanent tablespace for the database?

A. You'll first need to first reassign another tablespace as the new default (permanent) tablespace for the database.

B. The original tablespace will continue to be the default tablespace.

C. You must first remove all the current users from the tablespace.

D. You can't drop a default permanent tablespace.

13. Which one of the following is true regarding tablespace types?

A. If you create a database with bigfile set as the default tablespace type, you can't convert later to smallfile tablespaces.

B. If you create a database with smallfile set as the default tablespace type, you can't convert later to bigfile tablespaces.

C. It doesn't matter what default type of tablespace you create a database with; you can change the default tablespace type anytime you wish.

D. You can't switch the tablespace type once you create a database.

14. To avoid a large number of extents for bigfile tablespaces in very large databases, which of the following does Oracle recommend that you do?

A. Use the `UNIFORM` clause for allocating new extents and set a very high extent size.

B. Use the `AUTOALLOCATE` clause and simply let Oracle take care of the extent sizing.

C. Use the `AUTOALLOCATE` clause for allocating new extents and set a very high extent size.

D. Use the `UNIFORM` clause and simply let Oracle take care of the extent sizing.

Using Sorted Hash Clusters

15. Which of the following is true regarding creating tables for a sorted hash cluster?

A. You must first create the hash cluster before you create the member tables.

B. You must create the member tables before you create the cluster.

C. You create the tables and the cluster together, in the same SQL statement.

D. Once you create the clusters, the tables that are part of the cluster are created automatically by Oracle.

16. What happens if you use an `ORDER BY` clause on a suffix of the sort key columns?

A. You won't need to perform any sorting if you use an `ORDER BY` clause.

B. You'll need to perform additional sorting when you use an `ORDER BY` clause.

C. You'll perform the same amount of sorting as when there is no suffix on the sort key columns.

D. You cannot use an `ORDER BY` clause on a suffix of the sort key column.

Copying Files Using the Database Server

17. Which one of the following statements is true when you use the `DBMS_FILE_TRANSFER` package to transport files?

A. If the source file is a UNIX file, the destination file can be an ASM file.

B. If the destination file is a UNIX file, the source file can be a UNIX file.

C. If the destination file is a UNIX file, the source file can be an ASM file.

D. If the source file is an ASM file, the destination file can be an ASM file.

18. If you want to transfer a file using the `DBMS_FILE_TRANSFER` package, what do you need to do?

A. You must use the `COPY_FILE` procedure to copy files to a different server.

B. You must use the `COPY_FILE` procedure to copy files to a local database.

C. You must use the `PUT_FILE` procedure to copy files to a different server.

D. You must use the `GET_FILE` procedure to copy files to a different server.

LAB QUESTIONS

1. On the USERS tablespace, set a warning threshold of 80 percent and a critical threshold of 95 percent.

2. How do you check the database-wide threshold values for the USERS tablespace?

3. How do you turn off the space-usage tracking for the USER tablespace?

4. How do you reset the database-wide threshold values of the USERS tablespace to the default database values?

5. How do you check the status of your threshold?

SELF TEST ANSWERS

Proactive Tablespace Management

1. ☑ **A** and **D. A** is correct because you can provide alerts only in locally managed tablespaces. **D** is correct, since you can't provide alerts in an offline tablespace.
☒ **B** is wrong since dictionary-managed tablespaces aren't eligible for proactive tablespace alerts. **C** is a wrong answer because you can't provide alerts for read-only tablespaces.

2. ☑ **B.** You will get a maximum of one undo tablespace alert during a 24-hour period.
☒ **A, C,** and **D** provide wrong time intervals.

3. ☑ **B.** The MMON background process monitors the tablespace thresholds and is responsible for sending proactive alerts.
☒ **A, C,** and **D** don't send out any alerts.

Reclaiming Unused Space

4. ☑ **D.** You cannot shrink undo segments belonging to the undo tablespace.
☒ **A, B,** and **D** are wrong answers since they all point to eligible candidates for segment shrinking.

5. ☑ **D.** The release of the space would have an impact on DML operations.
☒ **A, B,** and **C** are wrong since the command would compact the data first and then release it, meanwhile impacting the DML operations going on in the database.

6. ☑ **B.** You must ensure that row movement is enabled, since the shrinking process could alter some ROWIDs in a heap-organized table.
☒ **A, C,** and **D** point out the wrong requirement.

Using the Undo and Redo Logfile Size Advisors

7. ☑ **A** and **C.** The Undo Advisor helps you set the undo retention period, as well as the size of the undo tablespace.
☒ **B** is wrong since there is no such thing as an undo interval. **D** is wrong because when you use Automatic Undo Management, you don't create the undo segments—Oracle does this.

8. ☑ **A.** The `FAST_START_MTTR_TARGET` initialization parameter sets the mean time to recovery (MTTR) following an instance crash.
☒ **B, C,** and **D** are wrong since the `FST_START_MTTR_TARGET` deals with instance recovery rather than database restore, number of redo log groups, or undo management.

9. ☑ **D**. Optimally sized redo log files should be the smallest consistent with keeping the checkpointing to a minimum amount.
☒ **A, B,** and **C** aren't logically correct answers, since they ignore the MTTR target.

Tablespace Enhancements

10. ☑ **B.** Oracle will create the SYSAUX tablespace in a default location, even if you fail to specify it. The same is true of the SYSTEM tablespace.
☒ **A** is wrong since database creation will succeed even when you don't specify a SYSTEM or SYSAUX tablespace. If you issue the `CREATE DATABASE` statement with no other clauses, Oracle will create a database with datafiles for the SYSTEM and SYSAUX tablespaces in system-determined default locations. **C** is wrong since Oracle doesn't create a SYSTEM tablespace in lieu of the SYSAUX tablespace. **D** is wrong since you can't create the SYSAUX tablespace after database creation—the SYSAUX tablespace is always created at database creation.

11. ☑ **B.** When you rename a default permanent tablespace, nothing really changes—the old default permanent tablespace continues to be the default permanent tablespace, with a new name.
☒ **A** is wrong since you don't need to create a new default permanent tablespace for the users. **C** is wrong since the users continue to be assigned to the same permanent tablespace throughout. **D** is wrong since you can rename a permanent tablespace.

12. ☑ **A.** You must ensure that there is always a default tablespace for all the users in the database, before you can drop an old one.
☒ **B** is wrong since the original default permanent tablespace will be replaced by the new default permanent tablespace. **C** is wrong since you don't remove any users from the tablespace. **D** is wrong since you can drop a default permanent tablespace after making sure you have an alternative default permanent tablespace for all the users.

13. ☑ **C.** It doesn't matter what type of tablespace you create the database with. You can always change back and forth between the two types of tablespaces: smallfile and bigfile.
☒ **A, B,** and **D** are wrong, since they contradict **C.**

14. ☑ **A.** Oracle recommends that you set the extent size yourself, by using the `UNIFORM` clause and selecting a very large extent size.
☒ **B** is wrong because you use the `AUTOALLOCATE` for normal sized, not very large databases. **C** is wrong for the same reason as **B**. Besides, you can't choose extent sizes when you use `AUTOALLOCATE`. **D** specifies the correct option, `UNIFORM`, but turns out to be the wrong answer, since it states that you should let Oracle choose the extent sizes.

Using Sorted Hash Clusters

15. ☑ **A.** The hash cluster must exist before you can create any member tables.
☒ **B** is wrong because you must first create the cluster and then the tables that are part of the cluster. **C** is wrong since you need to create the hash cluster first. **D** is wrong since Oracle doesn't automatically create the tables that are part of a cluster.

16. ☑ **B.** If you use an `ORDER BY` clause on a suffix of the sort key columns or nonsort key columns, additional sorting is required, assuming there are no indexes on the table.
☒ **A** is wrong since you'll need to perform sorting even with the `ORDER BY` clause. **C** is wrong since you'll perform more sorting than before. **D** is wrong since you *can* use an `ORDER BY` clause on a suffix of the sort key column.

Copying Files Using the Database Server

17. ☑ **B** and **D.** The source and destination files must be of the same type—UNIX or ASM.
☒ **A** and **C** are wrong, since the file systems don't match in both cases.

18. ☑ **B** and **C. B** is correct because the `COPY_FILE` procedure is meant for local file copies. **C** is correct because you must use the `PUT_FILE` procedure to transfer files to a remote database server.
☒ **A** is wrong because you can only make copies from a local server with the `COPY_FILE` procedure. **D** is wrong since the `GET_FILE` procedure helps you bring files from a remote server to your local server, not the other way around, as the answer indicates.

LAB ANSWERS

1. On the USERS tablespace, use the following to set a warning threshold of 80 percent and a critical threshold of 95 percent:

```
SQL> begin
       dbms_server_alert.set_threshold (
       dbms_server_alert.tablespace_pct_full,
       dbms_server_alert.operator_ge, 80,
       dbms_server_alert.operator_ge, 95, 1, 1, NULL,
       dbms_server_alert.object_type_tablespace, 'USERS');
      end;
```

You can use the NULL value to return to the database-wide default values.

2. Check the database-wide threshold values for the USERS tablespace as follows:

```
SQL> select warning_value, critical_value
     from dba_thresholds
     whre metrics_name = 'Tablespace Space Usage' AND
              object_name = 'USERS'
       /
```

3. Turn off the space-usage tracking for the USER tablespace as follows:

```
SQL> begin
       dbms_server_alert.set_threshold (
       dbms_server_alert.tablespace_pct_full,
       dbms_server_alert.operator_do_not_check, '0',
       dbms_server_alert.operator_do_not_check, '0', 1, 1, NULL,
       dbms_server_alert.object_type_tablespace, 'USERS');
     end;
```

4. Reset the database-wide threshold values of the USERS tablespace to the default database values as follows:

```
SQL> begin
       dbms_server_alert.set_threshold (
       dbms_server_alert.tablespace_pct_full,
       NULL, NULL, NULL, NULL, 1, 1, NULL,
       dbms_server_alert.object_type_tablespace, 'USERS');
     end;
```

5. Check the status of your threshold as follows:

```
SQL> select reason, resolution
     from dba_alert_history
     where object_name = 'USERS';
SQL> select reason, message_level
     from dba_outstanding_alerts
     where object_name = 'USERS';
```