5. Managing Tablespaces and Data Files.

Abstract: In this lesson you will discover in more details the physical and logical data storage entities envisioned in the Oracle server architecture. The previous lessons discussed the physical and logical structures, and two of the three components of the physical database structure — control files and redo log files. The third component of the physical structure is data files. Data files belong to logical units called tablespaces. In this lesson you will learn to manage data files and tablespaces..

Contents

Objectives:

- Describe the logical structure of tablespaces within the database
- Create tablespaces
- Change the size of the tablespace
- Allocate space for temporary segments
- Change the status of tablespaces
- Change the storage settings of tablespaces
- Implement Oracle Managed Files

1. Tablespaces and Data Files

The database's data is stored logically in *tablespaces* and physically in the *data files* corresponding to the tablespaces. The logical storage management is independent of the physical storage of the data files. A tablespace can have more than one data file associated with it. One data file belongs to only one tablespace. A database can have one or more tablespaces. Figure 1 shows the relationship between the database, tablespaces, data files, and the objects in the database. Any object (such as a non-partitioned table, an index, and so on) created in the database is stored in a single tablespace. But the object's physical storage can be on multiple data files belonging to that tablespace. A segment cannot be stored in multiple tablespaces.

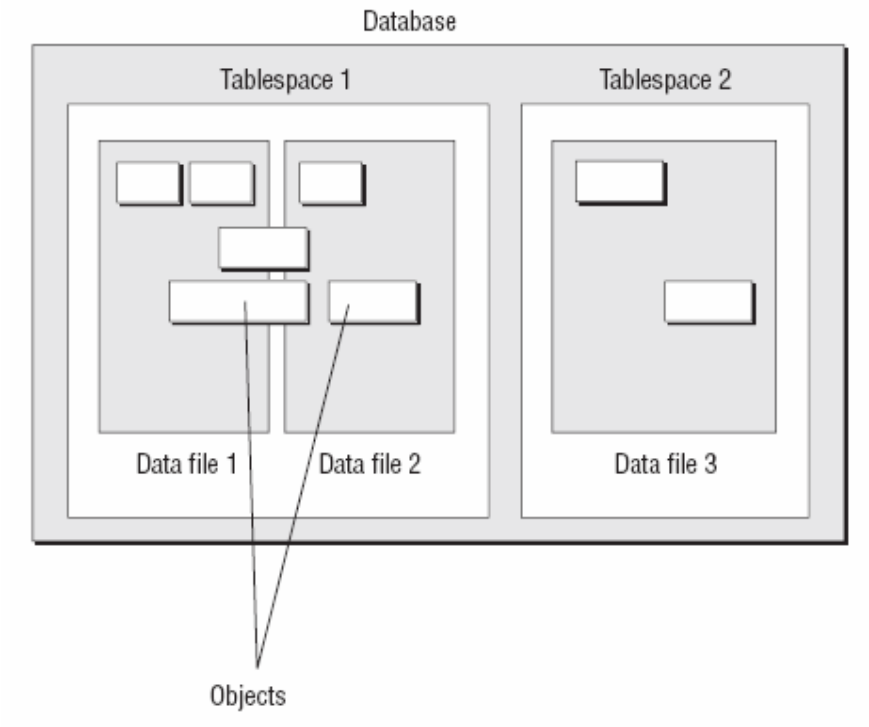

Fig. 1. Tablespaces and Data Files.

The size of the tablespace is the total size of all the data files belonging to that tablespace. The size of the database is the total size of all tablespaces in the database, which is the total size of all data files in the database. The smallest logical unit of storage in a database is a database block. You define the size of the block when you create the database, and you cannot alter it. The database block size is a multiple of the operating system block size.

Changing the size of the data files belonging to a tablespace can change the size of that tablespace. You can add more space to a tablespace by adding more data files to the tablespace. You can add more space to the database by adding more tablespaces, by adding more data files to the existing tablespaces, or by increasing the size of the existing data files.

When you create a database, Oracle creates the SYSTEM tablespace. All the dictionary objects are stored in this tablespace. The data files you specify when you create the database are assigned to the SYSTEM tablespace. You can add more space to the SYSTEM tablespace after you create the database by adding more data files or by increasing the size of the data files. The PL/SQL program units (such as procedures, functions, packages, or triggers) created in the database are also stored in the SYSTEM tablespace.

Oracle recommends not creating any objects other than the Oracle data dictionary in the SYSTEM tablespace. By having multiple tablespaces, you can do the following:

- Separate the Oracle dictionary from other database objects. Doing so reduces contention between dictionary objects and database objects for the same data file.
- Control I/O by allocating separate physical storage disks for different tablespaces.

- Manage space quotas for users on tablespaces.
- Have separate tablespaces for temporary segments (TEMP) and undo management (Rollback segments). You can also create a tablespace for a specific activity—for example, you can place high-update tables in a separate tablespace. When creating the database, you can specify tablespace names for temporary tablespace and undo tablespace.
- Group application-related or module-related data together, so that when maintenance is required for the application's tablespace, only that tablespace need be taken offline, and the rest of the database is available for users.
- Back up the database one tablespace at a time.
- Make part of the database read-only.

When you create a tablespace, Oracle creates the data files with the size specified. The space reserved for the data file is formatted but does not contain any user data. Whenever spaces for objects are needed, extents are allocated from this free space.

2. Managing Tablespaces

When Oracle allocates space to an object in a tablespace, it is allocated in chunks of contiguous database blocks known as extents. Each object is allocated a segment, which has one or more extents.

Oracle maintains the extent information such as extents free, extent size, extents allocated, and so on either in the data dictionary or in the tablespace itself.

If you store the *extent management* information in the dictionary for a tablespace, that tablespace is called a dictionary-managed tablespace.

Whenever an extent is allocated or freed, the information is updated in the corresponding dictionary tables. Such updates also generate undo information.

If you store the management information in a tablespace itself, by using bitmaps in each data file, such a tablespace is known as a locally managed tablespace. Each bit in the bitmap corresponds to a block or a group of blocks. When an extent is allocated or freed for reuse, Oracle changes the bitmap values to show the new status of the blocks. These changes do not

generate rollback information because they do not update tables in the data dictionary.

To create the tablespace, you use the CREATE TABLESPACE statement. You can modify the characteristics of the tablespace using the ALTER TABLESPACE statement. In the following sections, we'll create, modify, and drop a tablespace, and we'll query the tablespace information from the data dictionary.

Creating a Tablespace

As the database grows bigger, managing database objects is easier if you have multiple tablespaces. Using the CREATE TABLESPACE statement creates a tablespace. In Oracle9i, the only mandatory clause in the CREATE TABLESPACE statement is the tablespace name.

For example, when you specify the statement CREATE TABLESPACE APPLICATION_DATA, Oracle9i creates a locally managed tablespace with system allocated extent sizes.

The data file for this tablespace is created at the location you specify in the DB_CREATE_FILE_DEST parameter, and its size is 100MB. The file is auto extensible with no maximum size and has a name similar to ora_applicat_zyykpt00.dbf. In the following sections, we will discuss the default values for a CREATE TABLESPACE statement and the various types of tablespaces.

OBJECTIVE: CREATE TABLESPACES

TIP: The tablespace name cannot exceed 30 characters. The name should begin with an alphabetic character and can contain alphabetic characters, numeric characters, and the special characters #, _, and$.

Optionally, you can specify file names, file sizes, and default storage parameters when creating tablespaces. The default storage parameters are used whenever a new object is created (whenever a new segment is allocated) in the tablespace. The storage parameters you specify when you create an object override the default storage parameters of the tablespace containing the object. The default storage parameters for the tablespace are used only when you create an object without specifying any storage parameters. In Oracle9i, you can create tablespaces without specifying a file name by setting the parameter DB_CREATE_FILE_DEST to a valid directory on the server where you want to create the file. Files created in such manner are called Oracle Managed Files (OMF).

You can specify the extent management clause when creating a tablespace. If you do not specify the extent management clause, Oracle creates a locally managed tablespace. You can have both dictionary managed and locally managed tablespaces in the same database. A temporary tablespace can be either dictionary-managed or locally managed.

Dictionary-Managed Tablespaces

In dictionary-managed tablespaces, all extent information is stored in the data dictionary. A simple example of a dictionary-managed tablespace creation command is as follows:

```
CREATE TABLESPACE APPL_DATA
DATAFILE '/disk3/oradata/DB01/appl_data01.dbf' SIZE 100M;
EXTENT MANAGEMENT DICTIONARY;
```

This statement creates a tablespace named APPL_DATA; the data file specified is created with a size of 100MB. You can specify more than one file under the DATAFILE clause separated by commas; you may need to create more files if there are any operating system limits on the file size. For example, if you need to allocate 6GB for the tablespace, and the

operating system allows only a 2GB maximum, you need three data files for the tablespace. The statement is then as follows:

```
CREATE TABLESPACE APPL_DATA
DATAFILE '/disk3/oradata/DB01/appl_data01.dbf' SIZE 2000M,
        '/disk3/oradata/DB01/appl_data02.dbf' SIZE 2000M,
        '/disk4/oradata/DB01/appl_data03.dbf' SIZE 2000M;
```

The following statement creates a tablespace using all optional clauses.

```
CREATE TABLESPACE APPL_DATA
        DATAFILE '/disk3/oradata/DB-1/appl_data01.dbf' SIZE 100M
        DEFAULT STORAGE (
                INITIAL 256K
                NEXT 256K
                MINEXTENTS 2
                PCTINCREASE 0
                MAXEXTENTS 4096)
        BLOCKSIZE 4K
        MINIMUM EXTENT 256K
        LOGGING
        ONLINE
        PERMANENT
        EXTENT MANAGEMENT DICTIONARY
        SEGMENT SPACE MANAGEMENT MANUAL;
```

The clauses in the CREATE TABLESPACE command specify the following:

DEFAULT STORAGE Specifies the default storage parameters for new objects that are created in the tablespace. If you specify an explicit storage clause when creating an object, the tablespace defaults are not used for the specified storage parameters. You specify storage parameters within parentheses; no parameter is mandatory, but if you specify the DEFAULT STORAGE clause, you must specify at least one parameter inside the parentheses.

BLOCKSIZE Specifies the block size that is used for the objects created in the tablespace. By default, this block size is the database block size, which you define using the DB_BLOCK_SIZE parameter when creating the database. In Oracle9i, a database can have multiple block sizes. The database block size specified by DB_BLOCK_SIZE parameter is used for the SYSTEM tablespace and is known as the standard block size. The valid sizes of non-standard block size are 2KB, 4KB, 8KB, 16KB, and 32KB. If you do not specify a block size for the tablespace, the database block size is assumed. Multiple block sizes in the database are beneficial for large databases with OLTP (Online Transaction Processing) and DSS (Decision Support System) data stored together or for storing large tables.

In the "Using Non-standard Block Sizes" section, we'll discuss the restrictions on specifying non-standard block sizes when you create the tablespace.

INITIAL Specifies the size of the object's (segment's) first extent. NEXT specifies the size of the segment's next and successive extents. The size is specified in bytes. You can also specify the size in KB by post-fixing the size with K, or you can specify MB by post-fixing the size with M. The default value of INITIAL and NEXT is 5 database blocks. The minimum value of INITIAL is 3 database blocks for locally managed tablespaces (for manual segment space management, it is 2 blocks plus 1 block for each free list group in the segment) and 2 blocks for dictionary-managed tablespaces; NEXT is 1 database block. Even if you specify sizes smaller than these values, Oracle allocates the minimum sizes when creating segments in the tablespace.

PCTINCREASE Specifies how much the third and subsequent extents grow over the preceding extent. The default value is 50, meaning that each subsequent extent is 50 percent larger than the preceding extent. The minimum value is 0, meaning all extents after the first are the same size. For example, if you specify storage parameters as (INITIAL 1M NEXT 2M

PCTINCREASE 0), the extent sizes are 1MB, 2MB, 2MB, 2MB, and so on. If you specify the PCTINCREASE as 50, the extent sizes are 1MB, 2MB, 3MB, 4.5MB, 6.75MB, and so on. The actual NEXT extent size is rounded to a multiple of the block size.

MINEXTENTS Specifies the total number of extents allocated to the segment at the time of creation. Using this parameter, you can allocate a large amount of space when you create an object, even if the space available is not contiguous. The default and minimum value is 1. When you specify MINEXTENTS as more than 1, the extent sizes are calculated based on NEXT and PCTINCREASE. MAXEXTENTS specifies the maximum number of extents that can be allocated to a segment. You can specify an integer or UNLIMITED. The minimum value is 1, and the default value depends on the database block size.

MINIMUM EXTENT Specifies that the extent sizes are a multiple of the size specified. You can use this clause to control fragmentation in the tablespace by allocating extents of at least the size specified and as always a multiple of the size specified. In the CREATE TABLESPACE example earlier in this lesson, all the extents allocated in the tablespace are a multiple of 256KB. The INITIAL and NEXT extent sizes you specify should be a multiple of MINIMUM EXTENT.

LOGGING Specifies that the DDL operations and direct-load INSERT are recorded in the redo log files. LOGGING is the default, and you can omit the clause. When you specify NOLOGGING, data is modified with minimal logging and hence the commands complete faster. Since the changes are not recorded in the redo log files, you need to apply the commands again if you have to recover media. Specifying LOGGING or NOLOGGING in the individual object creation statement overrides the tablespace default.

ONLINE Specifies that the tablespace be created online or available as soon as it is created. ONLINE is the default, and hence you can omit the clause. If you do not want the tablespace to be available, you can specify OFFLINE.

PERMANENT Specifies whether the tablespace is to be used to create permanent objects such as tables, indexes, and so on. PERMANENT is the default, and hence you can omit it. If you plan to use the tablespace for temporary segments (such as to handle sorts in SQL), you

can mark the tablespace as TEMPORARY. You cannot create permanent objects such as tables or indexes in a TEMPORARY tablespace. We'll discuss temporary tablespaces later on.

EXTENT MANAGEMENT Until Oracle9i, dictionary managed tablespaces were the default. That is, if you did not specify the EXTENT MANAGEMENT clause, Oracle created a dictionary-managed tablespace. In Oracle9i, to create a dictionary-managed tablespace, you need to explicitly specify the EXTENT MANAGEMENT DICTIONARY clause. If you omit this clause, Oracle creates the tablespace as locally managed.

SEGMENT SPACE MANAGEMENT This clause is applicable only to locally managed tablespaces. The valid values are MANUAL and AUTO. MANUAL is the default. If you specify AUTO, Oracle manages the free space in the segments using bit maps rather than free lists. For AUTO, Oracle ignores the storage parameters PCTUSED, FREELISTS, and FREELIST GROUPS when creating objects.

Using Non-standard Block Sizes

When creating the database, you specify the block size in the initialization parameter using the DB_BLOCK_SIZE parameter. This specification is known as the standard block size for the database. You must choose a block size that suits most of your tables. In most databases, you will never need another block size. Oracle9i gives you option of using multiple block sizes, which is especially useful when you're transporting tablespaces from another database with a different block size. When creating a tablespace with a non-standard block size, you must specify the BLOCKSIZE clause in the CREATE TABLESPACE statement. You cannot alter this block size.

The DB_CACHE_SIZE parameter defines the buffer cache size associated with the standard block size. To create tablespaces with non-standard block size, you must set the appropriate initialization parameter to define a buffer cache size for the block size. The initialization parameter is DB_*n*K_CACHE_SIZE; n is the non-standard block size. It can have the values 2, 4, 8, 16, or 32, but cannot have the size of the standard block size. For example, if your standard block size is 8KB, you cannot set the parameter DB_8K_CACHE_SIZE. If you need to create a tablespace that uses a different block size, say 16KB, you must set the DB_16K_CACHE_SIZE parameter. By default, the value for DB_*n*K_CACHE_SIZE parameters is 0MB.

Temporary tablespaces should have standard block size.

> *TIP*: The DB_*n*K_CACHE_SIZE The parameter is dynamic; you can alter its value using the ALTER SYSTEM statement.

Locally Managed Tablespace

Using the CREATE TABLESPACE command with the EXTENT MANAGEMENT LOCAL clause creates a locally managed tablespace. Locally managed tablespaces manage space

more efficiently, provide better methods to reduce fragmentation, and increase reliability. The extent allocation information is stored as bitmaps in the file headers, and hence it improves the speed of allocation and deallocation operations. You cannot specify the DEFAULT STORAGE, TEMPORARY, and MINIMUM EXTENT clauses of the CREATE TABLESPACE in a locally managed tablespace.

You can specify that Oracle manage extents automatically by using the AUTOALLOCATE option. When using this option, you cannot specify sizes for the objects created in the tablespace. Oracle manages the extent sizes; you have no control over the extent sizes or the extent's allocation and deallocation.

Following is an example of creating a locally managed tablespace with Oracle managing the extent allocation (since EXTENT MANAGEMENT LOCAL AUTOALLOCATE is the default, it is omitted from the statement):

```
CREATE TABLESPACE USER_DATA
DATAFILE '/disk1/oradata/MYDB01/user_data01.dbf' SIZE 300M;
```

You can specify that the tablespace be managed with uniform extents of a specific size by using the UNIFORM SIZE clause. All the extents will be created with the size you specify. You cannot specify extent sizes (STORAGE clause) when creating the tablespace. The following is an example of creating a locally managed tablespace with uniform extent sizes of 512KB.

```
CREATE TABLESPACE USER_DATA
DATAFILE '/disk1/oradata/MYDB01/user_data01.dbf' SIZE 300M
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 512K;
```

If you set the DB_CREATE_FILE_DEST parameter to a valid directory on the server and execute the statement CREATE TABLESPACE MYTS, a new tablespace MYTS is created as locally managed with auto allocated extent sizes, standard block size, and manual segment space management.

You manage the free space in the tablespace's segment using free lists. In Oracle9i, the database can manage the free space by using bitmaps. If you want a locally managed tablespace to have the segment's free space managed by Oracle, you must specify the SEGMENT SPACE MANAGEMENT AUTO clause in the CREATE TABLESPACE statement.

Free Space Management

When creating locally managed permanent tablespaces, you can specify the SEGMENT SPACE MANAGEMENT clause to AUTO or MANUAL. MANUAL is the default and is the only behavior available in pre-Oracle9i databases.

In pre-Oracle9i databases, you managed the free space in the segments by using free lists.

In Oracle9i, Oracle can manage the free space in blocks using bitmaps if you specify the SEGMENT SPACE MANAGEMT AUTO clause while creating a tablespace. A bitmap that shows the status of the block is maintained—the status shows whether a block is available for insert. Thus, performance improves when multiple sessions are doing inserts to the same block. Space is also used effectively for objects with varying size rows.

When you set SEGMENT SPACE MANAGEMENT to AUTO, for all segments created in the tablespace Oracle ignores the storage parameters FREELISTS, FREELIST GROUPS, and PCTUSED. Therefore, you need not worry about tuning these parameters!

Undo Tablespace

Oracle9i can manage undo information automatically. You need not create rollback segments and worry about their sizes and number. For automatic undo management, you must have one undo tablespace. You create the undo tablespace using the CREATE UNDO TABLESPACE statement. When creating undo tablespace, you can specify only the EXTENT MANAGEMENT LOCAL and DATAFILE clauses. Oracle creates a locally managed permanent tablespace. The following statement creates an undo tablespace:

```
CREATE UNDO TABLESPACE UNDO_TBS
DATAFILE '/ora1/oradata/MYDB/undo_tbs01.dbf' SIZE 500M;
```

TIP: You can create an undo tablespace when creating a database using the UNDO TABLESPACE clause of the CREATE DATABASE statement.

Temporary Tablespace

OBJECTIVE: Allocate space for temporary segments

Oracle can manage space for sort operations more efficiently by using temporary tablespaces. By exclusively designating a tablespace for temporary segments, Oracle eliminates allocation and deallocation of temporary segments. A temporary tablespace can be used only for sort segments. Only one sort segment is allocated for an instance in a temporary tablespace, and all sort operations use this sort segment. More than one transaction can use the same sort segment, but each extent can be used by only one transaction.

The sort segment for a given temporary tablespace is created at the time of the first sort operation on that tablespace. The sort segment expands by allocating extents until the segment size is sufficient for the total storage demands of all the active sorts running on that instance.

A temporary tablespace can be dictionary-managed or locally managed. Using the CREATE TABLESPACE command with the TEMPORARY clause creates a dictionary-managed temporary tablespace. Here is an example:

```
CREATE TABLESPACE TEMP
DATAFILE '/disk5/oradata/MYDB01/temp01.dbf' SIZE 300M
DEFAULT STORAGE (INITIAL 2M NEXT 2M PCTINCREASE 0
      MAXEXTENTS UNLIMITED)
TEMPORARY;
```

When the first sort operation is performed on disk, a temporary segment is allocated with a 2MB initial extent size and 2MB subsequent extent sizes. The extents, once allocated, are freed only when the instance is shut down.

Temporary segments are based on the default storage parameters of the tablespace. For a TEMPORARY tablespace, the recommended INITIAL and NEXT parameters should be equal, and the extent size should be a multiple of SORT_AREA_SIZE plus DB_BLOCK_SIZE to reduce the possibility of fragmentation. Keep PCTINCREASE equal to zero. For example, if your sort area size is 64KB and database block size is 8KB, provide the default storage of the temporary tablespace as (INITIAL 136K NEXT 136K PCTINCREASE 0 MAXEXTENTS UNLIMITED).

If you are using a PERMANENT tablespace for sort operations, temporary segments are created in the tablespace when the sort is performed and are freed when the sort operation completes. There will be one sort segment for each sort operation, which requires a lot of extent and segment management operations.

To create a locally managed temporary tablespace, use the CREATE TEMPORARY TABLESPACE command. The following statement creates a locally managed temporary tablespace:

```
CREATE TEMPORARY TABLESPACE TEMP
TEMPFILE '/disk5/oradata/MYDB01/temp01.tmp' SIZE 500M
EXTENT MANAGEMENT LOCAL
UNIFORM SIZE 5M;
```

Notice that the DATAFILE clause of the CREATE TABLESPACE command is replaced with the TEMPFILE clause. Temporary files are always in NOLOGGING mode and are not recoverable. They cannot be made read-only, cannot be renamed, cannot be created with the ALTER DATABASE command, do not generate any information during the BACKUP CONTROLFILE command, and are not included during a CREATE CONTROLFILE command. The EXTENT MANAGEMENT LOCAL clause is optional and can be omitted; it is provided to improve readability. If you do not specify the extent size by using the UNIFORM SIZE clause, the default size used is 1MB.

NOTE: An Oracle temporary file (called a tempfile) is not a temporary file in the traditional operating system sense; only the objects within a temporary tablespace consisting of one or more tempfiles are temporary.

When you create a user, that user is assigned a temporary tablespace. By default, the default tablespace (where the user creates objects) and the temporary tablespace (where the user's sort operations are performed) are both the SYSTEM tablespace. No user should have SYSTEM as their default or temporary tablespace. This unnecessarily increases fragmentation in the SYSTEM tablespace.

When creating a database, you can also create a temporary tablespace using the DEFAULT TEMPORARY TABLESPACE clause of the CREATE DATABASE statement. If the default temporary tablespace is defined in the database, all new users will have that tablespace assigned as the temporary tablespace by default, if you do not specify another tablespace for the user's temporary tablespace.

Altering a Tablespace

You can alter a tablespace using the ALTER TABLESPACE command. This command allow you to do the following:

- Change the default storage parameters of a dictionary-managed tablespace
- Change the extent allocation and LOGGING/NOLOGGING modes
- Change the tablespace from PERMANENT to TEMPORARY or vice versa
- Change the availability of the tablespace
- Make the tablespace read-only or read-write
- Coalesce the contiguous free space
- Add more space by adding new data files or temporary files
- Rename files belonging to the tablespace
- Begin and end a backup

OBJECTIVE: Change the storage settings of tablespaces

Changing the default storage, extent allocation, or LOGGING/NOLOGGING does not affect the existing objects in the tablespace. The DEFAULT STORAGE and LOGGING/NOLOGGING clauses are applied to the newly created segments if you do not explicitly define such a clause when creating new objects. For example, to change the storage parameters, use the following statement:

```
ALTER TABLESPACE APPL_DATA
DEFAULT STORAGE (INITIAL 2M NEXT 2M);
```

Only the INITIAL and NEXT values of the storage clause are changed; the other storage parameters such as PCTINCREASE or MINEXTENTS remain unaltered.

You can change a dictionary-managed temporary tablespace to permanent or vice versa by using the ALTER TABLESPACE command, if the tablespace is empty and the permanent tablespace uses standard block size.

You cannot use the ALTER TABLESPACE command, with the TEMPORARY keyword, to change a locally managed permanent tablespace into a locally managed temporary tablespace. You must use the CREATE TEMPORARY TABLESPACE statement to create a locally managed temporary tablespace. However, you can use the ALTER TABLESPACE command to change a locally managed temporary tablespace to a locally managed permanent tablespace. The following statement changes a tablespace to temporary:

```
ALTER TABLESPACE TEMP TEMPORARY;
```

TIP: The clauses in the ALTER TABLESPACE command are all mutually exclusive; you can specify only one clause at a time.

WARNING: You cannot use the ALTER TABLESPACE statement to change the tablespace's extent allocation to DICTIONARY or LOCAL.

Tablespace Availability

OBJECTIVE: Change the status of tablespaces

You can control the availability of certain tablespaces by placing them offline or online. When you make a tablespace offline, the segments in that tablespace are not accessible. The data stored in other tablespaces is available for use. When making a tablespace unavailable, you can use the following four options:

NORMAL This option is the default. Oracle writes all the dirty buffer blocks in the SGA (System Global Area) to the data files of the tablespace and closes the data files. All data files belonging to the tablespace must be online. You need not do a media recovery when bringing the tablespace online. For example:

```
ALTER TABLESPACE USER_DATA OFFLINE NORMAL
```

TEMPORARY Oracle performs a checkpoint on all online data files. It does not ensure that the data files are available. You might need to perform a media recovery on the offline data files when the tablespace is brought online. For example:

```
ALTER TABLESPACE USER_DATA OFFLINE TEMPORARY;
```

IMMEDIATE Oracle does not perform a checkpoint and does not make sure that all data files are available. You must perform a media recovery when the tablespace is brought back online. For example:

```
ALTER TABLESPACE USER_DATA OFFLINE IMMEDIATE;
```

FOR RECOVER This option places the tablespace offline for point-in-time recovery. You can copy the data files belonging to the tablespace from a backup and apply the archive log files. For example:

```
ALTER TABLESPACE USER_DATA OFFLINE FOR RECOVER;
```

This option is deprecated since Oracle9i, and is available only for backward compatibility.

You cannot place the SYSTEM tablespace offline because the data dictionary must always be available for the database to function. If a tablespace is offline when you shut down the database, it remains offline when you start up the database. You can place a tablespace offline by using the statement

```
ALTER TABLESPACE USER_DATA ONLINE.
```

When you take a tablespace offline, SQL statements cannot reference any objects contained in that tablespace. If there are unsaved changes when you take the tablespace offline, Oracle saves rollback data corresponding to those changes in a deferred rollback segment in the SYSTEM tablespace.

When you bring the tablespace back online, Oracle applies the rollback data to the tablespace, if needed.

Coalescing Free Space

You can use the ALTER TABLESPACE command with the COALESCE clause to coalesce the adjacent free extents. When you free up the extents used by an object, either by altering the object storage or by dropping the object, Oracle does not combine the adjacent free extents. When coalescing tablespaces, Oracle does not combine all free space into one big extent; Oracle combines only the adjacent extents. For example, Figure 2 shows the extent allocation in a tablespace before and after coalescing.

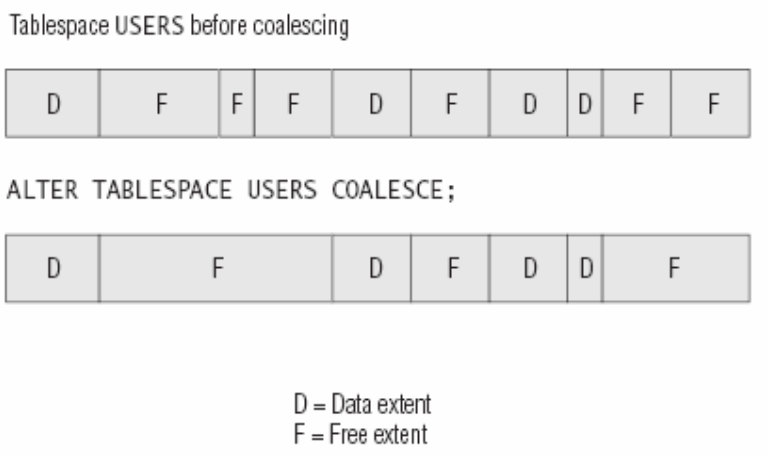

Fig. 2. Coalescing a tablespace.

The SMON (system monitor) process coalesces the tablespace. If you set the PCTINCREASE storage parameter for the tablespace to a nonzero value, the SMON process automatically coalesces the tablespace's unused extents. Even if you set PCTINCREASE to zero, Oracle coalesces the tablespace when it does not find a free extent that is big enough. Oracle also does a limited amount of coalescing if the PCTINCREASE value of the object dropped is not zero. If the extent sizes of the tablespace are all uniform, there is no need to coalesce.

Read-Only Tablespace

If you do not want users to change any data in the tablespace, you can specify that it is read-only. All objects in the tablespace are available for queries. INSERT, UPDATE, and DELETE operations on the data are not allowed. When the tablespace is made read-only, the data file headers are no longer updated when the checkpoint occurs. You need to back up the *read-only tablespaces* only once. You cannot make the SYSTEM tablespace read-only. When you make a tablespace read-only, all the data files must be online, and the tablespace can have no pending transactions.

You can drop objects such as tables or indexes from a read-only tablespace, but you cannot create new objects in a read-only tablespace.

To make the USERS tablespace read-only, use the following statement:

```
ALTER TABLESPACE USERS READ ONLY;
```

If you issue the ALTER TABLESPACE <*tablespace name*> READ ONLY statement when there are active transactions in the tablespace, the tablespace goes into a transitional read-only mode in which no further DML (Data Manipulation Language) statements are allowed, although existing transactions that are modifying the tablespace are allowed to commit or roll back.

To change a tablespace to read-write mode, use the following command:

```
ALTER TABLESPACE USERS READ WRITE;
```

Oracle normally checks the availability of all data files belonging to the database when starting up the database. If you are storing your read-only tablespace on an offline storage medium or on a CD-ROM, you might want to skip the data file availability checking when starting up the database. To do so, set the parameter READ_ONLY_OPEN_DELAYED to TRUE. Oracle checks the availability of data files belonging to read-only tablespaces only at the time of access to an object in the tablespace. A missing or bad read-only file will not be detected at database start-up time.

Adding Space to a Tablespace

OBJECTIVE: Change the size of the tablespace

You can add more space to a tablespace by adding more data files to it or by changing the size of the existing data files. To add more data files or temporary files to the tablespace, use the ALTER TABLESPACE command with the ADD [DATAFILE/TEMPFILE] clause. For example, to add a file to a tablespace, run the following command:

```
ALTER TABLESPACE USERS ADD DATAFILE
    '/disk5/oradata/DB01/users02.dbf' SIZE 25M;
```

If you are modifying a locally managed temporary tablespace to add more files, use the following statement:

```
ALTER TABLESPACE USER_TEMP ADD TEMPFILE
    '/disk4/oradata/DB01/user_temp01.dbf' SIZE 100M;
```

For locally managed temporary tablespaces, you can use only the ADD TEMPFILE clause with the ALTER TABLESPACE command.

Dropping a Tablespace

You use the DROP TABLESPACE statement to drop a tablespace from the database. If the tablespace to be dropped is not empty, use the INCLUDING CONTENTS clause. For example, to drop the USER_DATA tablespace, use the following statement:

```
DROP TABLESPACE USER_DATA;
```

If the tablespace is not empty, specify the following:

```
DROP TABLESPACE USER_DATA INCLUDING CONTENTS;
```

If there are referential integrity constraints from the objects on other tablespaces referring to the objects in the tablespace that is being dropped, you must specify the CASCADE CONSTRAINTS clause:

```
DROP TABLESPACE USER_DATA INCLUDING CONTENTS CASCADE
CONSTRAINTS;
```

When you drop a tablespace, the control file is updated with the tablespace and data file information. The actual data files belonging to the tablespace are removed only if the data files are Oracle Managed Files.

If the files are not Oracle managed, you can either use operating system commands to remove the data files belonging to the dropped tablespace or use the AND DATAFILES clause to free up the disk space. The following statement drops the tablespace and removes all data files belonging to the tablespace from the disk.

```
DROP TABLESPACE USER_DATA INCLUDING CONTENTS AND DATAFILES;
```

You cannot drop the SYSTEM tablespace.

Querying Tablespace Information

You query tablespace information from the following data dictionary views.

DBA_TABLESPACES

The DBA_TABLESPACES view shows information about all tablespaces in the database. (USER_TABLESPACES shows tablespaces that are accessible to the user.) This view contains default storage parameters and specifies the type of tablespace, the status, and so on

```
SQL> SELECT TABLESPACE_NAME, EXTENT_MANAGEMENT,
  2     ALLOCATION_TYPE, CONTENTS,
  3     SEGMENT_SPACE_MANAGEMENT
  4     FROM DBA_TABLESPACES;

TABLESPACE_NAME      EXTENT_MAN     ALLOCATIO CONTENTS  SEGMEN
---------------      ----------     --------- --------- ------
SYSTEM     DICTIONARY      USER      PERMANENT      MANUAL
UNDOTBS    LOCAL           SYSTEM    UNDO           MANUAL
```

```
CWMLITE     LOCAL       SYSTEM    PERMANENT   MANUAL
DRSYS       LOCAL       SYSTEM    PERMANENT   MANUAL
EXAMPLE     LOCAL       SYSTEM    PERMANENT   MANUAL
TEMP        LOCAL       UNIFORM   TEMPORARY   MANUAL
TOOLS       LOCAL       SYSTEM    PERMANENT   MANUAL
USERS       LOCAL       SYSTEM    PERMANENT   MANUAL
APP_DATA    DICTIONARY  USER      PERMANENT   MANUAL
APP_INDEX   LOCAL       UNIFORM   PERMANENT   AUTO
10 rows selected.
SQL>
```

The following columns are displayed in DBA_TABLESPACES view:

TABLESPACE_NAME
BLOCK_SIZE
INITIAL_EXTENT
NEXT_EXTENT
MIN_EXTENTS
MAX_EXTENTS
PCT_INCREASE
MIN_EXTLEN
STATUS
CONTENTS
LOGGING
EXTENT_MANAGEMENT
ALLOCATION_TYPE
PLUGGED_IN
SEGMENT_SPACE_MANAGEMENT

V$_TABLESPACE

V$TABLESPACE shows the tablespace number, the name, and the backup status from the control file.

```
SQL> SELECT * FROM V$TABLESPACE;
       TS#        NAME                           INC
---------- ------------------------------ -----
         2      CWMLITE                        YES
         3      DRSYS                          YES
```

```
         4 EXAMPLE                        YES
        11 APP_INDEX                      YES
         0 SYSTEM                         YES
         7 TOOLS                          YES
         1 UNDOTBS                        YES
         8 USERS                          YES
         6 TEMP                           YES
        10 APP_DATA                       YES
10 rows selected.
SQL>
```

DBA_FREE_SPACE

This view shows the free extents available in all tablespaces. You use this view to find the total free space available in a tablespace. USER_FREE_SPACE shows the free extents in tablespaces accessible to the current user. Locally managed temporary tablespaces are not shown in this view.

```
SQL> SELECT TABLESPACE_NAME, SUM(BYTES) FREE_SPACE
  2    FROM DBA_FREE_SPACE
  3    GROUP BY TABLESPACE_NAME;

TABLESPACE_NAME                FREE_SPACE
------------------------------ ----------
APP_DATA                       10481664
APP_INDEX                      10223616
CWMLITE                        14680064
DRSYS                          12845056
EXAMPLE                        196608
SYSTEM                         88281088
TOOLS                          4390912
UNDOTBS                        208338944
USERS                          24051712
9 rows selected.
SQL>
```

The following columns are displayed in DBA_FREE_SPACE view:

TABLESPACE_NAME
FILE_ID

```
BLOCK_ID
BYTES
BLOCKS
RELATIVE_FNO
```

V$SORT_USAGE

This view shows information about the active sorts in the database; it shows the space used, the username, the SQL address, and the hash value. You can join this view with V$SESSION or V$SQL to get more information about the session.

```
SQL> SELECT USER, SESSION_ADDR, SESSION_NUM, SQLADDR,
  2    SQLHASH, TABLESPACE, EXTENTS, BLOCKS
  3    FROM V$SORT_USAGE;

USER        SESSION_  SESSION_NUM    SQLADDR
----------  --------  -------------  --------
      SQLHASH   TABLESPACE       EXTENTS    BLOCKS
      ----------  ------------------  ----------  ----------
SCOTT       030539F4  24                  0343E200
      1877781575  TEMP                45          360
```

The following columns are displayed in V$SORT_USAGE view:

```
USERNAME
USER
SESSION_ADDR
SESSION_NUM
SQLADDR
SQLHASH
TABLESPACE
CONTENTS
SEGTYPE
SEGFILE#
SEGBLK#
EXTENTS
BLOCKS
SEGRFNO#
```

Other Views

The following views also show information related to tablespaces:

DBA_SEGMENTS, **USER_SEGMENTS** Shows information about the segments, segment types, size, and storage parameter values associated with tablespaces. This example shows the tablespace and total space occupied by each type of segment owned by PM schema.

```
SQL> SELECT TABLESPACE_NAME, SEGMENT_TYPE, SUM(BYTES)
  2     FROM DBA_SEGMENTS
  3     WHERE OWNER = 'PM'
  4     GROUP BY ROLLUP(TABLESPACE_NAME, SEGMENT_TYPE);

TABLESPACE_NAME      SEGMENT_TYPE       SUM(BYTES)
-------------------- ------------------ -------------
EXAMPLE              INDEX              196608
EXAMPLE              LOBINDEX           1114112
EXAMPLE              LOBSEGMENT         1572864
EXAMPLE              NESTED TABLE       65536
EXAMPLE              TABLE              131072
EXAMPLE                                 3080192
                                        3080192

7 rows selected.
SQL>
```

DBA_EXTENTS, **USER_EXTENTS** Shows information about the extents, extent sizes, associated segment, and tablespace.

DBA_DATA_FILES Shows data files belonging to tablespaces.

DBA_TEMP_FILES Shows temporary files belonging to locally managed temporary tablespaces.

V$TEMP_EXTENT_MAP Shows all extents of a locally managed temporary tablespace.

V$TEMP_EXTENT_POOL Shows the temporary space used and cached for the current instance, for locally managed temporary tablespaces.

V$TEMP_SPACE_HEADER Shows the space used and free in each temporary tablespace files.

V$SORT_SEGMENT Shows information about sort segments.

DBA_USERS Shows information about the default and temporary tablespace assignments to users. The following query shows the default tablespace assignments of user HR.

```
SQL> SELECT DEFAULT_TABLESPACE, TEMPORARY_TABLESPACE
```

```
  2    FROM DBA_USERS
  3    WHERE USERNAME = 'HR';

DEFAULT_TABLESPACE             TEMPORARY_TABLESPACE
------------------------------ ----------------------
EXAMPLE                        TEMP
SQL>
```

3. Managing Data Files

Data files (or temporary files) are created when you create a tablespace or when you alter a tablespace to add files. Before Oracle9i, you had to specify a file name and size to create files. Since Oracle9i, Oracle can create files and remove them when the tablespace is removed. Such files are known as Oracle Managed Files. We'll look at how you can use OMF to specify data files later in this lesson.

You can specify the size of the file when you create a file or reuse an existing file. When you reuse an existing file, that file should not belong to any Oracle database—the contents of the file are overwritten. Use the REUSE clause to specify an existing file. If you omit the REUSE clause and the data file being created exists, Oracle returns an error. For example:

```
CREATE TABLESPACE APPL_DATA
DATAFILE '/disk2/oradata/DB01/appl_data01.dbf' REUSE;
```

When you specify REUSE, you can omit the SIZE clause. If you specify the SIZE clause, the size of the file should be the same as the existing file. If the file to be created does not exist, Oracle creates a new file even if you specify the REUSE clause.

Always specify the fully qualified directory name for the file being created. If you omit the directory, Oracle creates the file under the default database directory or in the current directory, depending on the operating system.

In the following sections we will discuss how to specify file sizes, resize data files, relocate tablespaces by renaming the data files, and display the dictionary views that contain information about data files and temporary files.

Sizing Files

OBJECTIVE: Change the size of the tablespace

You can specify that the data file (or temporary file) grow automatically whenever space is needed in the tablespace. To do so, specify the AUTOEXTEND clause for the file. This functionality enables you to have fewer data files per tablespace and can simplify the administration of data files. You can turn the AUTOEXTEND clause ON and OFF; you can also specify file size increments. You can set a maximum limit for the file size; by default the file size limit is UNLIMITED. You can specify the AUTOEXTEND clause for files when you run the CREATE DATABASE, CREATE TABLESPACE, ALTER TABLESPACE, or ALTER DATAFILE commands. For example:

```
CREATE TABLESPACE APPL_DATA
DATAFILE '/disk2/oradata/DB01/appl_data01.dbf'
SIZE 500M
AUTOEXTEND ON NEXT 100M MAXSIZE 2000M;
```

The AUTOEXTEND ON clause specifies that the automatic file resize feature be enabled for the specified file; NEXT specifies the size by which the file should be incremented; and MAXSIZE specifies the maximum size for the file. When Oracle tries to allocate an extent in the tablespace, it looks for a free extent. If Oracle cannot locate a large enough free extent (even after coalescing), Oracle increases the data file size by 100MB and tries to allocate the new extent.

The following statement disables the automatic file extension feature:

```
ALTER DATABASE
DATAFILE '/disk2/oradata/DB01/appl_data01.dbf'
AUTOEXTEND OFF;
```

If the file already exists in the database, and you want to enable the auto extension feature, use the ALTER DATABASE command. For example, you can use the following statement:

```
ALTER DATABASE
DATAFILE '/disk2/oradata/DB01/appl_data01.dbf'
AUTOEXTEND ON NEXT 100M MAXSIZE 2000M;
```

You can increase or decrease the size of a data file or temporary file (thus increasing or decreasing the size of the tablespace) by using the RESIZE clause of the ALTER DATABASE DATAFILE command. For example, to redefine the size of a file, use the following statement:

```
ALTER DATABASE
DATAFILE '/disk2/oradata/DB01/appl_data01.dbf'
RESIZE 1500M;
```

When decreasing the file size, Oracle returns an error if it finds data beyond the new file size. You cannot reduce the file size below the high-water mark in the file. Reducing the file size helps to reclaim unused space.

Oracle Managed Files

OBJECTIVE: Implement Oracle Managed Files

Oracle managed files are appropriate for smaller non-production databases or databases on disks that use the logical volume manager (LVM). LVM is software available with most disk systems to combine partitions of multiple physical disks to one logical volume. LVM can use mirroring, striping, RAID 5, and so on. The following benefits are associated with using the OMF:

Prevention of errors Because Oracle removes the files associated with the tablespace, you cannot make a mistake by removing a file that belongs to an active tablespace.

Standard naming convention The files you create using OMF have unique and standard file names.

Space retrieval When tablespaces are removed, Oracle removes the files associated with the tablespace, thus freeing up space immediately on the disk. The DBA might forget to remove the file from disk.

Easy script writing Application vendors need not worry about the syntax of specifying directory names in the scripts when porting an application to multiple platforms. The same script can be used to create tablespaces on different operating system platforms.

You can use OMF to create files and to remove them when the corresponding object (redo log group or tablespace) is dropped from the database. You manage OMF-created files, using the traditional methods for renaming or resizing files.

Creating Files

Before you can create Oracle Managed Files, you must set the parameter DB_CREATE_FILE_DEST. You can specify this parameter in the initialization parameter file or set/change it using the ALTER SYSTEM or ALTER SESSION statement. The DB_CREATE_FILE_DEST parameter defines the directory where Oracle can create data files. Oracle must have read/write permission on this directory, and the directory must exist on the server where the database is located. Oracle will not create the directory; it will create only create the data file.

You can use OMF to create data files when using the CREATE DATABASE, CREATE TABLESPACE, or ALTER TABLESPACE statements. In the CREATE DATABASE statement, you need not specify the data file names for SYSTEM or UNDO or TEMPORARY

tablespaces. You can omit the DATAFILE clause in the CREATE TABLESPACE statement. In the ALTER TABLESPACE ADD DATAFILE statement, you can omit the file name.

The data files you create using OMF will have a standard format. For a data file the format is ora_%t_%u.dbf. The format for a temp file is ora_%t_%u.tmp; %t is the tablespace name, and %u is a unique 8-character string that Oracle derives. If the tablespace name is more than 8 characters, only the first 8 characters are used. The file names that Oracle generates are reported in the alert log file.

You can also use the OMF feature to create control files and redo log files of the database. Since these two types of files can be multiplexed, Oracle provides another parameter to specify the location of files — DB_CREATE_ONLINE_LOG_DEST_*n*, in which *n* can be 1, 2, 3, 4, or 5. You can also alter these initialization parameters using ALTER SYSTEM or ALTER SESSION. If you set the parameters DB_CREATE_ONLINE_LOG_DEST_1 and DB_CREATE_ONLINE_LOG_DEST_2 in the parameter file when creating a database, Oracle creates two control files (one in each directory) and creates two online redo log groups with two members each (one member each in both directories).

The redo log file names will have the format ora_%g_%u.log, in which %g is the log group number and %u is an 8-character string unique to the database. The control file name will have the format ora_%u.ctl, in which %u is an 8-character string.

Let's consider an example of creating a database. The following parameters are set in the initialization parameter file.

```
UNDO_MANAGEMENT = AUTO
DB_CREATE_ONLINE_LOG_DEST_1 = '/ora1/oradata/MYDB'
DB_CREATE_ONLINE_LOG_DEST_2 = '/ora2/oradata/MYDB'
DB_CREATE_FILE_DEST = '/ora1/oradata/MYDB'
```

The CONTROL_FILES parameter is not set. Create the database using the following statement:

```
CREATE DATABASE MYDB
DEFAULT TEMPORARY TABLESPACE TEMP;
```

The following files will be created.:

- The SYSTEM tablespace data file in /ora1/oradata/MYDB
- The TEMP tablespace temp file in /ora1/oradata/MYDB
- A control file in /ora1/oradata/MYDB
- A control file in /ora2/oradata/MYDB
- One member of the first redo log group in /ora1/oradata/MYDB and a second member in /ora2/oradata/MYDB
- One member of the second redo log group in /ora1/oradata/MYDB and a second member in /ora2/oradata/MYDB

Because we specified the UNDO_MANAGEMENT clause and did not specify a name for the undo tablespace, Oracle creates SYS_UNDOTBS tablespace as undo tablespace and creates its data file under /ora1/oradata/MYDB. If you omit the DEFAULT TEMPORARY TABLESPACE clause, Oracle will not create a temporary tablespace.

NOTE: When using OMF to create control files, you must get the names of control files from the alert log and add them to the initialization parameter file using the CONTROL_FILES parameter, for the instance to start again.

WARNING: If you do not specify the DB_CREATE_ONLINE_LOG_DEST_*n* parameter when creating a database or when adding a redo log group, OMF creates one control file and two groups with one member each for redo log files in the DB_CREATE_FILE_DEST directory. If you also do not set the DB_CREATE_FILE_DEST parameter and you did not provide data file names and redo log file names, Oracle creates the files under a default directory (usually $ORACLE_HOME/dbs), but these files will not be Oracle managed. This is the default behavior of the database.

The data files and temp files that OMF creates will have a default size of 100MB, which is auto extensible with no maximum file size. Each redo log member will be 100MB in size by default.

Let's look at another example that creates two tablespaces. The data file for the APP_DATA tablespace will be stored in the /ora5/oradata/MYDB directory. The data file for the APP_INDEX tablespace will be stored in the /ora6/oradata/MYDB directory.

```
ALTER SESSION SET DB_CREATE_FILE_DEST = '/ora5/oradata/MYDB';
CREATE TABLESPACE APP_DATA
EXTENT MANAGEMENT DICTIONARY;
ALTER SESSION SET DB_CREATE_FILE_DEST = '/ora6/oradata/MYDB';
CREATE TABLESPACE APP_INDEX;
```

Overriding the Default File Size

If you want a different size for the files created by OMF, you can specify the DATAFILE clause without a file name. You can also turn off the auto-extensible feature of the data file. The following statement creates a tablespace of 10MB and turns off the auto-extensible feature:

```
CREATE TABLESPACE PAY_DATA DATAFILE SIZE 10M
AUTOEXTEND OFF;
```

Here is another example, which creates multiple data files for the tablespace. The second and third data files are auto-extensible.

```
CREATE TABLESPACE PAY_INDEX
DATAFILE SIZE 20M AUTOEXTEND OFF,
SIZE 30M AUTOEXTEND ON MAXSIZE 1000M,
SIZE 1M;
```

The following example adds files to an existing tablespace.

```
ALTER SYSTEM SET DB_CREATE_FILE_DEST = '/ora5/oradata/MYDB';
ALTER TABLESPACE USERS ADD DATAFILE;
ALTER SYSTEM SET DB_CREATE_FILE_DEST = '/ora8/oradata/MYDB';
ALTER TABLESPACE APP_DATA
ADD DATAFILE SIZE 200M AUTOEXTEND OFF;
```

NOTE: Once created, Oracle-managed files are treated as other database files. You can rename and resize them, and you must back them up. Archive log files are not OMF.

REAL WORLD SCENARIO

How Do You Create a Database and Tablespaces with OMF?

Your manager has asked you to create a test database for a new application your company just bought. The database is for testing the functionality of the application. The vendor told you that you need four tablespaces: SJC_DATA, SJC_INDEX, WKW_DATA, and WKW_INDEX. The index tablespaces must have uniform extent sizes of 512KB and should have minimum size of 500MB. The SJC_DATA tablespace is to be dictionary managed with the minimum and extent size multiple to be 128K and a tablespace size of 1GB. The SJC_DATA tablespace should be 250MB.

Since this is a database for testing the functionality of the application, you decide to use Oracle Managed Files, which makes your life easier by creating and cleaning the files in the database.

Let's create the database. Your systems administrator has given you four disks: /ora1, /ora2, /ora3, and /ora4, each with 900MB of space.

Be sure to include the following in the parameter file:

```
UNDO_MANAGEMENT = AUTO
DB_CREATE_FILE_DEST = /ora1
DB_CREATE_ONLINE_LOG_DEST_1 = /ora1
DB_CREATE_ONLINE_LOG_DEST_2 = /ora2
```

Create the database using the following statement:

```
CREATE DATABASE SJCTEST
LOGFILE SIZE 20M
DEFAULT TEMPORARY TABLESPACE TEMP
TEMPFILE SIZE 200M
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 2M
UNDO TABLESPACE UNDO_TBS SIZE 200M;
```

This statement creates a database named SJCTEST. The system tablespace, undo tablespace, and temporary tablespace are created in /ora1. System tablespace has the default size of 100M; undo tablespace and temporary tablespace have the size of 200MB. Since we did not want each log file member to be 100MB, we specified a smaller size for online redo log members.

Two control files and redo log files with two members are created. Each member is stored in /ora1 and /ora2.

After running the necessary scripts to create the catalog and packages, we'll create the tablespaces for the application.

```
ALTER SYSTEM SET DB_CREATE_FILE_DEST = "/ora2";
CREATE TABLESPACE SJC_DATA
EXTENT MANAGEMENT DICTIONARY
MINIMUM EXTENT 128K
DATAFILE SIZE 800M;
ALTER SYSTEM SET DB_CREATE_FILE_DEST = "/ora3";
ALTER TABLESPACE SJC_DATA ADD DATAFILE SIZE 200M;
CREATE TABLESAPCE WKW_INDEX
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 512K
DATAFILE SIZE 500M;
ALTER SYSTEM SET DB_CREATE_FILE_DEST = "/ora4";
CREATE TABLESPACE WKW_DATA;
CREATE TABLESPACE SJC_INDEX
EXTENT MANAGEMENT LOCAL UNIFORM SIZE 512K
DATAFILE SIZE 500M;
```

Since we have only 900MB in each file system, we need two data files to allocate 1GB to the SJC_DATA tablespace. This is accomplished in two steps.

Renaming and Relocating Files

You rename data files using the RENAME FILE clause of the ALTER DATABASE command. You can also rename data files by using the RENAME DATAFILE clause of the ALTER TABLESPACE command. You use the RENAME functionality to logically move tablespaces from one location to another. Consider the following example.

The tablespace USER_DATA has three data files:

- /disk1/oradata/DB01/user_data01.dbf
1. /disk1/oradata/DB01/userdata2.dbf
2. /disk1/oradata/DB01/user_data03.dbf

You'll notice that the second file does not follow the naming standard set for your company, so you need to rename the file. Follow these steps:

1. Take the tablespace offline.

   ```
   ALTER TABLESPACE USER_DATA OFFLINE;
   ```

2. Copy or move the file to the new location, or rename the file by using operating system commands.
3. Rename the file in the database by using one of the following two commands:

   ```
   ALTER DATABASE RENAME FILE
   '/disk1/oradata/DB01/userdata2.dbf' TO
   '/disk1/oradata/DB01/user_data02.dbf';
   ```

   or

   ```
   ALTER TABLESPACE USER_DATA RENAME DATAFILE
   '/disk1/oradata/DB01/userdata2.dbf' TO
   '/disk1/oradata/DB01/user_data02.dbf';
   ```

4. Bring the tablespace online.

   ```
   ALTER TABLESPACE USER_DATA ONLINE;
   ```

If you need to relocate the tablespace from disk 1 to disk 2, follow the same steps. You can rename all the files in the tablespace by using a single command. The steps are as follows:

1. Take the tablespace offline.

   ```
   ALTER TABLESPACE USER_DATA OFFLINE;
   ```

2. Copy the file to the new location by using operating system commands on the disk.
3. Rename the files in the database by using one of the following two commands. The number of data files specified before the keyword TO should equal the number of files specified after the keyword.

```
ALTER DATABASE RENAME FILE
'/disk1/oradata/DB01/user_data01.dbf',
'/disk1/oradata/DB01/userdata2.dbf',
'/disk1/oradata/DB01/user_data03.dbf'
TO
'/disk2/oradata/DB01/user_data01.dbf',
'/disk2/oradata/DB01/user_data02.dbf',
'/disk2/oradata/DB01/user_data03.dbf';
```

or

```
ALTER TABLESPACE USER_DATA RENAME DATAFILE
'/disk1/oradata/DB01/user_data01.dbf',
'/disk1/oradata/DB01/userdata2.dbf',
'/disk1/oradata/DB01/user_data03.dbf'
TO
'/disk2/oradata/DB01/user_data01.dbf',
'/disk2/oradata/DB01/user_data02.dbf',
'/disk2/oradata/DB01/user_data03.dbf';
```

4. Bring the tablespace online.

```
ALTER TABLESPACE USER_DATA ONLINE;
```

To rename or relocate files that belong to multiple tablespaces, or if the file belongs to the SYSTEM tablespace, follow these steps:

1. Shut down the database. A complete backup is recommended before making any structural changes.
2. Copy or rename the files on the disk by using operating system commands.
3. Start up and mount the database (STARTUP MOUNT).
4. Rename the files in the database by using the ALTER DATABASE RENAME FILE command.
5. Open the database by using ALTER DATABASE OPEN.

If you need to move read-only tablespaces to CD-ROM or to any write-once read-many device, follow these steps:

1. Make the tablespace read-only.
2. Copy the data files that belong to the tablespace to the device.
3. Rename the files in the database by using the ALTER DATABASE RENAME FILE command.

Querying Data File Information

You can query data file and temporary file information by using the following views.

V$DATAFILE

This view shows data file information from the control file.

```
SQL> SELECT FILE#, RFILE#, STATUS, BYTES, BLOCK_SIZE
  2    FROM V$DATAFILE;

     FILE#      RFILE#     STATUS     BYTES      BLOCK_SIZE
----------  ----------  ---------- ----------  ----------
         1           1  SYSTEM     267386880   8192
         2           2  ONLINE     545259520   8192
         3           3  ONLINE     17039360    8192
         4           4  ONLINE     75497472    8192
         5           5  ONLINE     17825792    8192
         6           6  ONLINE     26214400    8192
         7           7  ONLINE     92274688    8192
         8           8  ONLINE     31465472    8192
```

The following columns are displayed in V$DATAFILE view:

FILE#
CREATION_CHANGE#
CREATION_TIME
TS#
RFILE#
STATUS
ENABLED
CHECKPOINT_CHANGE#
CHECKPOINT_TIME
UNRECOVERABLE_CHANGE#
UNRECOVERABLE_TIME
LAST_CHANGE#
LAST_TIME
OFFLINE_CHANGE#
ONLINE_CHANGE#

```
ONLINE_TIME
BYTES
BLOCKS
CREATE_BYTES
BLOCK_SIZE
NAME
PLUGGED_IN
BLOCK1_OFFSET
AUX_NAME
```

V$TEMPFILE

Similar to the V$DATAFILE view, this view shows information about the temporary files.

```
SQL> SELECT FILE#, RFILE#, STATUS, BYTES, BLOCK_SIZE
  2    FROM V$TEMPFILE;

     FILE#      RFILE#     STATUS     BYTES      BLOCK_SIZE
----------  ----------  -------  ----------  ----------
         1           1  ONLINE     10485760        8192
```

DBA_DATA_FILES

This view shows information about the data file names, associated tablespace names, size, status, and so on.

```
SQL> SELECT TABLESPACE_NAME, FILE_NAME, BYTES,
  2    AUTOEXTENSIBLE
  3    FROM DBA_DATA_FILES;

TABLESPACE_NAME      FILE_NAME                              BYTES      AUT
-------------------- -------------------------------------- ---------- ---
SYSTEM               C:\ORACLE\BTWIN01\SYSTEM01.DBF         340787200  YES
APP_DATA             C:\ORACLE\ORA_APP_DATA_ZZPSNR00.       10485760   YES
                     DBF
UNDOTBS              C:\ORACLE\BTWIN01\UNDOTBS01.DBF        209715200  YES
CWMLITE              C:\ORACLE\BTWIN01\CWMLITE01.DBF        20971520   YES
DRSYS                C:\ORACLE\BTWIN01\DRSYS01.DBF          20971520   YES
EXAMPLE              C:\ORACLE\BTWIN01\EXAMPLE01.DBF        160563200  YES
```

```
APP_INDEX       C:\ORACLE\ORA_APP_INDE_ZZPT8F00.    10485760    YES
                DBF
TOOLS           C:\ORACLE\BTWIN01\TOOLS01.DBF       10485760    YES
USERS           C:\ORACLE\BTWIN01\USERS01.DBF       26214400    YES
PAY_DATA        C:\ORACLE\ORA_PAY_DATA_ZZQBLZ00.    10485760    NO
                DBF
PAY_INDEX       C:\ORACLE\ORA_PAY_INDE_ZZQBPF00.    20971520    NO
                DBF
PAY_INDEX       C:\ORACLE\ORA_PAY_INDE_ZZQBPG00.    31457280    NO
                DBF
```

The following columns are displayed in DBA_DATA_FILES view:

FILE_NAME
FILE_ID
TABLESPACE_NAME
BYTES
BLOCKS
STATUS
RELATIVE_FNO
AUTOEXTENSIBLE
MAXBYTES
MAXBLOCKS
INCREMENT_BY
USER_BYTES
USER_BLOCKS

DBA_TEMP_FILES

This view shows information similar to that of the DBA_DATA_FILES view for the temporary files in the database.

```
SQL> SELECT TABLESPACE_NAME, FILE_NAME, BYTES,
  2    AUTOEXTENSIBLE
  3    FROM DBA_TEMP_FILES;

TABLESPACE       FILE_NAME                          BYTES       AUT
---------------- ---------------------------------- ----------- ---------
TEMP_LOCAL       C:\ORACLE\DB01\TEMP_LOCAL01.DBF    10485760    NO
```

NOTE: The maximum number of data files per tablespace is depends on the operating system, but on most operating systems, it is 1022. The maximum number of data files per database is 65,533. The MAXDATAFILES clause in the CREATE DATABASE or CREATE CONTROLFILE statements also limits the number of data files per database. The maximum data file size is also depends on the operating system. There is no limit on the number of tablespaces per database. Because only 65,533 data files are allowed per database, you cannot have more than 65,533 tablespaces, because each tablespace needs at least one data file.

Summary

This lesson discussed the tablespaces and data files—the logical storage structures and physical storage elements of the database. A data file belongs to one tablespace, and a tablespace can have one or more data files.

The size of the tablespace is the total size of all the data files belonging to that tablespace. The size of the database is the total size of all tablespaces in the database, which is the same as the total size of all data files in the database.

Tablespaces are logical storage units used to group data by their type or category.

You create tablespaces using the CREATE TABLESPACE command. Oracle always allocates space to an object in chunks of blocks known as extents.

Tablespaces can handle the extent management through the Oracle dictionary or locally in the data files that belong to the tablespace. When creating tablespaces, you can specify default storage parameters for the objects that will be created in the tablespace. If you do not specify any storage parameters when creating an object, the storage parameters for the tablespace are used for the new object.

Locally managed tablespaces can have uniform extent sizes, which reduces fragmentation and wasted space. You can also specify that Oracle do the entire extent sizing for locally managed tablespaces.

A temporary tablespace is used only for sorting; you can't create permanent objects in a temporary tablespace. Only one sort segment is created for each instance in the temporary tablespace. Multiple transactions can use the same sort segment, but one transaction can use only one extent. To create locally managed temporary tablespaces, you use the CREATE TEMPORARY TABLESPACE command. Temporary files (instead of data files) are created when you use this command. Although these files are part of the database, they do not appear in the control file, and the block changes do not generate any redo information because all the segments created on locally managed temporary tablespaces are temporary segments.

You can alter a tablespace to change its availability or to make it readonly. Data in an offline tablespace is not accessible, whereas data in the read-only tablespaces cannot be modified or deleted. You can drop objects from a read-only tablespace.

Space is added to the tablespace by adding new data files to the tablespace or by increasing the size of the data files. You can obtain tablespace information from the dictionary using the

DBA_TABLESPACES and V$TABLESPACE views. The data files can be renamed through Oracle. This feature is useful to relocate a tablespace.

Oracle9i can manage the physical files belonging to the database using the Oracle Managed Files feature. OMF is good for non-production databases and databases on Logical Volume Manager. The V$DATAFILE, V$TEMPFILE, DBA_DATA_FILES, and DBA_TEMP_FILES views provide information on the data files.

References

[1] Oracle9i DBA Fundamentals I.