3. Database Structures

Abstract: In this lecture the students discover the basic mechanisms of any database system: the dictionary, control and log files, tablespaces, segments, storage structures, data blocks, extents, segments, the undo mechanisms.

Contents

Objective:

- Describe the mechanism involved with data blocks, extents, and segments
- Identify key data dictionary components, as well as the contents and uses of the data dictionary
- Describe tablespaces, the primary logical database structures of any database, and the physical datafiles that correspond to each tablespace.

3.1. Data Blocks, Extents, and Segments

Oracle allocates logical database space for all data in a database. The units of database space allocation are data blocks, extents, and segments. Figure 1 shows the relationships among these data structures.

At the finest level of granularity, Oracle stores data in **data blocks** (also called **logical blocks**, Oracle blocks, or pages). One data block corresponds to a specific number of bytes of physical database space on disk.

The next level of logical database space is an **extent**. An extent is a specific number of contiguous data blocks allocated for storing a specific type of information.

The level of logical database storage above an extent is called a **segment**. A segment is a set of extents, each of which has been allocated for a specific data structure and all of which are stored in the same tablespace. For example, each table's data is stored in its own **data segment**, while each index's data is stored in its own **index segment**. If the table or index is partitioned, each partition is stored in its own segment.

Oracle allocates space for segments in units of one extent. When the existing extents of a segment are full, Oracle allocates another extent for that segment. Because extents are allocated as needed, the extents of a segment may or may not be contiguous on disk.

A segment and all its extents are stored in one tablespace. Within a tablespace, a segment can include extents from more than one file; that is, the segment can span datafiles. However, each extent can contain data from only one datafile.

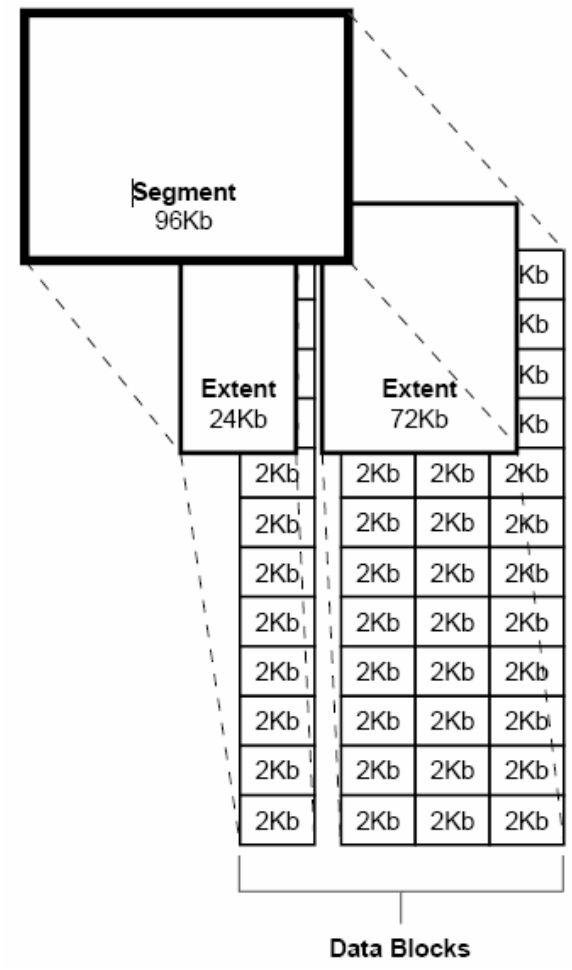

Figure 1. The Relationships Among Segments, Extents, and Data Blocks.

Although you can allocate additional extents, the blocks themselves are allocated separately. If you allocate an extent to a specific instance, the blocks are immediately allocated to the free list. However, if the extent is not allocated to a specific instance, then the blocks themselves are allocated only when the high water mark moves. The **high water mark** is the boundary between used and unused space in a segment.

Data Blocks Overview

Oracle manages the storage space in the datafiles of a database in units called **data blocks**. A data block is the smallest unit of data used by a database. In contrast, at the physical, operating system level, all data is stored in bytes. Each operating system has a **block size**. Oracle requests data in multiples of Oracle data blocks, not operating system blocks.

The standard block size is specified by the initialization parameter DB_BLOCK_SIZE. In addition, you can specify of up to five nonstandard block sizes. The data block sizes should be a multiple of the operating system's block size within the maximum limit to avoid unnecessary I/O. Oracle data blocks are the smallest units of storage that Oracle can use or allocate.

Data Block Format

The Oracle data block format is similar regardless of whether the data block contains table, index, or clustered data. Figure 2 illustrates the format of a data block.

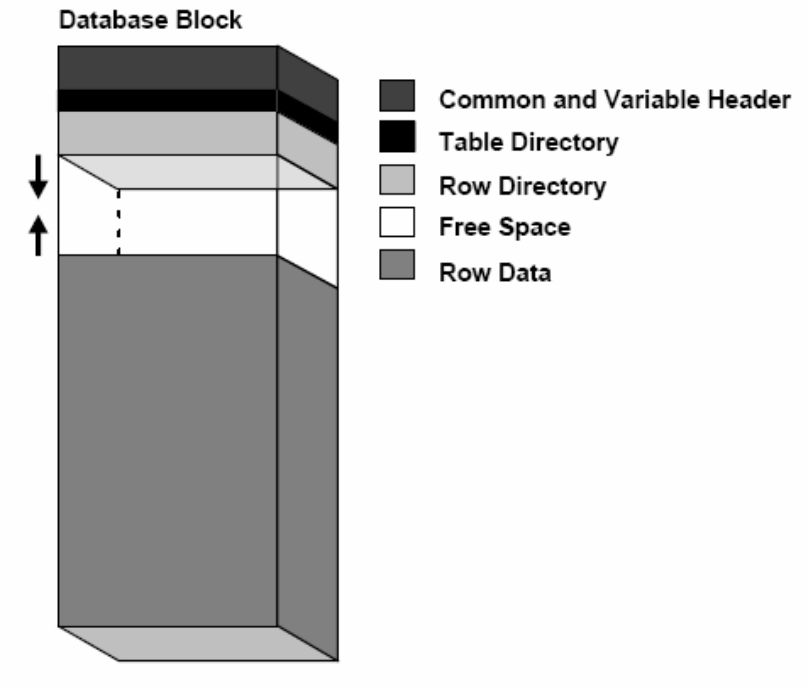

Figure 2. Data Block Format.

Header (Common and Variable)

The header contains general block information, such as the block address and the type of segment (for example, data or index).

Table Directory

This portion of the data block contains information about the table having rows in this block.

Row Directory

This portion of the data block contains information about the actual rows in the block (including addresses for each row piece in the row data area).

After the space has been allocated in the row directory of a data block's overhead, this space is not reclaimed when the row is deleted. Therefore, a block that is currently empty but had up to 50 rows at one time continues to' have 100 bytes allocated in the header for the row directory. Oracle reuses this space only when new rows are inserted in the block.

Overhead

The data block header, table directory, and row directory are referred to collectively as overhead. Some block overhead is fixed in size; the total block overhead size is variable. On average, the fixed and variable portions of data block overhead total 84 to 107 bytes.

Row Data

This portion of the data block contains table or index data. Rows can span blocks.

Free Space

Free space is allocated for insertion of new rows and for updates to rows that require additional space (for example, when a trailing null is updated to a non-null value).Whether issued insertions actually occur in a given data block is a function of current free space in that data block and the value of the space management parameter PCTFREE.

In data blocks allocated for the data segment of a table or cluster, or for the index segment of an index, free space can also hold transaction entries. A transaction entry is required in a block for each INSERT, UPDATE, DELETE, and SELECT...FOR UPDATE statement accessing one or more rows in the block. The space required for transaction entries is operating system dependent; however, transaction entries in most operating systems require approximately 23 bytes.

Free Space Management

Free space can be managed automatically or manually.

Free space can be managed automatically inside database segments. The in-segment free/used space is tracked using bitmaps, as opposed to free lists. Automatic segment-space management offers the following benefits:

- Ease of use
- Better space utilization, especially for the objects with highly varying size rows
- Better run-time adjustment to variations in concurrent access
- Better multi-instance behavior in terms of performance/space utilization

You specify automatic segment-space management when you create a locally managed tablespace. The specification then applies to all segments subsequently created in this tablespace.

Availability and Compression of Free Space in a Data Block

Two types of statements can increase the free space of one or more data blocks: DELETE statements, and UPDATE statements that update existing values to smaller values. The released space from these types of statements is available for subsequent INSERT statements under the following conditions:

- If the INSERT statement is in the same transaction and subsequent to the statement that frees space, then the INSERT statement can use the space made available.

- If the INSERT statement is in a separate transaction from the statement that frees space (perhaps being run by another user), then the INSERT statement can use the space made available only after the other transaction commits and only if the space is needed.

Released space may or may not be contiguous with the main area of free space in a data block. Oracle coalesces the free space of a data block only when (1) an INSERT or UPDATE statement attempts to use a block that contains enough free space to contain a new row piece, and (2) the free space is fragmented so the row piece cannot be inserted in a contiguous section of the block. Oracle does this compression only in such situations, because otherwise the performance of a database system decreases due to the continuous compression of the free space in data blocks.

Row Chaining and Migrating

In two circumstances, the data for a row in a table may be too large to fit into a single data block. In the first case, the row is too large to fit into one data block when it is first inserted. In this case, Oracle stores the data for the row in a chain of data blocks (one or more) reserved for that segment. Row chaining most often occurs with large rows, such as rows that contain a column of datatype LONG or LONG RAW. Row chaining in these cases is unavoidable.

However, in the second case, a row that originally fit into one data block is updated so that the overall row length increases, and the block's free space is already completely filled. In this case, Oracle migrates the data for the entire row to a new data block, assuming the entire row can fit in a new block. Oracle preserves the original row piece of a migrated row to point to the new block containing the migrated row. The rowid of a migrated row does not change.

When a row is chained or migrated, I/O performance associated with this row decreases because Oracle must scan more than one data block to retrieve the information for the row.

PCTFREE, PCTUSED, and Row Chaining

For manually managed tablespaces, two space management parameters, PCTFREE and PCTUSED, enable you to control the use of free space for inserts and updates to the rows in all the data blocks of a particular segment. Specify these parameters when you create or alter a table or cluster (which has its own data segment). You can also specify the storage parameter PCTFREE when creating or altering an index (which has its own index segment).

BLOCKSIZE

The BLOCKSIZE clause is used to specify the block size for the tablespace. If you don't specify this clause while tablespace creation then standard that is default blocksize is used which is specified by parameter DB_BLOCK_SIZE. For example, You have DB_BLOCK_SIZE set to 8K=8192 and you specified CREATE TABLESPACE command without any BLOCKSIZE clause then database use 8k blocksize for the specified tablespace.

The parameter DB_BLOCK_SIZE specifies the size of Oracle database blocks in bytes. The default value of this parameter is 8192 and value ranges between 2048 and 32768. But it must be multiple of physical block size at device level.

Now let's look at the DB_nK_CACHE_SIZE parameters. Here n is variable and can be 2, 4, 8, 16, 32. That is DB_2K_CACHE_SIZE, DB_4K_CACHE_SIZE, DB_16K_CACHE_SIZE etc are available. Now question may come why we will set DB_nK_CACHE_SIZE? This parameter needs to be set whenever one wishes to make or want to make a tablespace with non-standard data block size.

Suppose your standard block size set by DB_BLOCK_SIZE is 8K and you want to make a tablespace with block size 16K then at first you need to set DB_16K_CACHE_SIZE and then you need to create tablespace with the BLOCKSIZE clause specifying 16K.

Here is an example which shows database standard block size is 8k and you have made an tablespace with 16k blocksize.

Example:

SQL> show parameter db_block_size
NAME TYPE VALUE
---------- ----------- ----------
db_block_size integer 8192

ALTER SYSTEM SET DB_16K_CACHE_SIZE=200M SCOPE=BOTH;
ALTER SYSTEM SET DB_CREATE_FILE_DEST='/oradata2';
CREATE TABLESPACE TEST BLOCKSIZE 16K;

You can set this parameter only when DB_BLOCK_SIZE has a value other than nK. For example, if DB_BLOCK_SIZE=8192, then it is illegal to specify the parameter DB_8K_CACHE_SIZE because the size for the 8 KB block cache is already specified by DB_CACHE_SIZE.

Extents Overview

An extent is a logical unit of database storage space allocation made up of a number of contiguous data blocks. One or more extents in turn make up a segment. When the existing space in a segment is completely used, Oracle allocates a new extent for the segment.

When Extents Are Allocated

When you create a table, Oracle allocates to the table's data segment an initial extent of a specified number of data blocks. Although no rows have been inserted yet, the Oracle data blocks that correspond to the initial extent are reserved for that table's rows.

If the data blocks of a segment's initial extent become full and more space is required to hold new data, Oracle automatically allocates an incremental extent for that segment. An incremental extent is a subsequent extent of the same or greater size than the previously allocated extent in that segment.

For maintenance purposes, the header block of each segment contains a directory of the extents in that segment.

Determine the Number and Size of Extents

Storage parameters expressed in terms of extents define every segment. Storage parameters apply to all types of segments. They control how Oracle allocates free database space for a given segment. For example, you can determine how much space is initially reserved for a table's data segment or you can limit the number of extents the table can allocate by specifying the storage parameters of a table in the STORAGE clause of the CREATE TABLE statement. If you do not specify a table's storage parameters, then it uses the default storage parameters of the tablespace.

Prior to Oracle8i, all tablespaces were created as dictionary managed. Dictionary managed tablespaces rely on data dictionary tables to track space utilization. Beginning with Oracle8i, you could create locally-managed tablespaces, which use bitmaps (instead of data dictionary tables) to track used and free space. Because of the better performance and greater ease of management of locally managed tablespaces, the default for non-SYSTEM permanent tablespaces is locally managed whenever the type of extent management is not explicitly specified.

A tablespace that manages its extents locally can have either uniform extent sizes or variable extent sizes that are determined automatically by the system. When you create the tablespace, the UNIFORM or AUTOALLOCATE (system-managed) clause specifies the type of allocation.

- For system-managed extents, you can specify the size of the initial extent and Oracle determines the optimal size of additional extents, with a minimum extent size of 64 KB. This is the default for permanent tablespaces.

- For uniform extents, you can specify an extent size or use the default size, which is 1 MB. Temporary tablespaces that manage their extents locally can only use this type of allocation.

The storage parameters NEXT, PCTINCREASE, MINEXTENTS, MAXEXTENTS, and DEFAULT STORAGE are not valid for extents that are managed locally.

How Extents Are Allocated

Oracle uses different algorithms to allocate extents, depending on whether they are locally managed or dictionary managed.

With locally managed tablespaces, Oracle looks for free space to allocate to a new extent by first determining a candidate datafile in the tablespace and then searching the datafile's bitmap for the required number of adjacent free blocks. If that datafile does not have enough adjacent free space, thenOracle looks in another datafile.

When Extents Are Deallocated

In general, the extents of a segment do not return to the tablespace until you drop the schema object whose data is stored in the segment (using a DROP TABLE or DROP CLUSTER statement). Exceptions to this include the following:

- The owner of a table or cluster, or a user with the DELETE ANY privilege, can truncate the table or cluster with a TRUNCATE...DROP STORAGE statement.

- A database administrator (DBA) can deallocate unused extents using the following SQL syntax:

```
ALTER TABLE table_name DEALLOCATE UNUSED;
```

- Periodically, Oracle deallocates one or more extents of a rollback segment if it has the OPTIMAL size specified.

When extents are freed, Oracle modifies the bitmap in the datafile (for locally managed tablespaces) or updates the data dictionary (for dictionary managed tablespaces) to reflect the regained extents as available space. Any data in the blocks of freed extents becomes inaccessible.

Segments Overview

A segment is a set of extents that contains all the data for a specific logical storage structure within a tablespace. For example, for each table, Oracle allocates one or more extents to form that table's data segment, and for each index, Oracle allocates one or more extents to form its index segment.

Oracle databases use four types of segments:

- Data Segments
- Index Segments
- Temporary Segments

Introduction to Data Segments

A single data segment in an Oracle database holds all of the data for one of the following:

- A table that is not partitioned or clustered
- A partition of a partitioned table
- A cluster of tables

Oracle creates this data segment when you create the table or cluster with the CREATE statement.

The storage parameters for a table or cluster determine how its data segment's extents are allocated. You can set these storage parameters directly with the appropriate CREATE or ALTER statement. These storage parameters affect the efficiency of data retrieval and storage for the data segment associated with the object.

NOTE:

Oracle creates segments formaterialized views and materialized view logs in the same manner as for tables and clusters.

Introduction to Index Segments

Every non-partitioned index in an Oracle database has a single index segment to hold all of its data. For a partitioned index, every partition has a single index segment to hold its data.

Oracle creates the index segment for an index or an index partition when you issue the CREATE INDEX statement. In this statement, you can specify storage parameters for the extents of the index segment and a tablespace in which to create the index segment. (The segments of a table and an index associated with it do not have to occupy the same tablespace.) Setting the storage parameters directly affects the efficiency of data retrieval and storage.

Introduction to Temporary Segments

When processing queries, Oracle often requires temporary workspace for intermediate stages of SQL statement parsing and execution. Oracle automatically allocates this disk space called a temporary segment. Typically, Oracle requires a temporary segment as a work area for sorting. Oracle does not create a segment if the sorting operation can be done in memory or if Oracle finds some other way to perform the operation using indexes.

Operations that Require Temporary Segments

The following statements sometimes require the use of a temporary segment:

- CREATE INDEX
- SELECT ... ORDER BY
- SELECT DISTINCT ...
- SELECT ... GROUP BY
- SELECT ... UNION
- SELECT ... INTERSECT
- SELECT ... MINUS

Some unindexed joins and correlated subqueries can require use of a temporary segment. For example, if a query contains a DISTINCT clause, a GROUP BY, and an ORDER BY, Oracle can require as many as two temporary segments. If applications often issue statements in the previous list, the database administrator can improve performance by adjusting the initialization parameter SORT_AREA_SIZE.

Segments in Temporary Tables and Their Indexes

Oracle can also allocate temporary segments for temporary tables and indexes created on temporary tables. Temporary tables hold data that exists only for the duration of a transaction or session.

How Temporary Segments Are Allocated

Oracle allocates temporary segments differently for queries and temporary tables.

Allocation of Temporary Segments for Queries Oracle allocates temporary segments as needed during a user session in the temporary tablespace of the user issuing the statement. Specify this tablespace with a CREATE USER or an ALTER USER statement using the TEMPORARY TABLESPACE clause.

If no temporary tablespace is defined for the user, then the default temporary tablespace is the SYSTEM tablespace. The default storage characteristics of the containing tablespace determine those of the extents of the temporary segment. Oracle drops temporary segments when the statement completes.

Because allocation and deallocation of temporary segments occur frequently, create a special tablespace for temporary segments. By doing so, you can distribute I/O across disk devices, and you

can avoid fragmentation of the SYSTEM and other tablespaces that otherwise hold temporary segments.

> *NOTE:*
>
> When the SYSTEM tablespace is locally managed, you must define a default temporary tablespace when creating a database. A locally managed SYSTEM tablespace cannot be used for default temporary storage.

Entries for changes to temporary segments used for sort operations are not stored in the redo log, except for space management operations on the temporary segment.

Allocation of Temporary Segments for Temporary Tables and Indexes Oracle allocates segments for a temporary table when the first INSERT into that table is issued. (This can be an internal insert operation issued by CREATE TABLE AS SELECT.) The first INSERT into a temporary table allocates the segments for the table and its indexes, creates the root page for the indexes, and allocates any LOB segments.

Segments for a temporary table are allocated in the temporary tablespace of the user who created the temporary table.

Oracle drops segments for a transaction-specific temporary table at the end of the transaction and drops segments for a session-specific temporary table at the end of the session. If other transactions or sessions share the use of that temporary table, the segments containing their data remain in the table.

Automatic Undo Management

Automatic undo management is undo-tablespace based. You allocate space in the form of a few undo tablespaces, instead of allocating many rollback segments in different sizes.

Automatic undo management lets you explicitly control undo retention. Through the use of a system parameter (UNDO_RETENTION), you can specify the amount of committed undo information to retain in the database. You specify the parameter as clock time (for example, 30 seconds). With retention control, you can configure your system to enable long queries to run successfully.

Use the V$UNDOSTAT view to monitor and configure your database system to achieve efficient use of undo space. V$UNDOSTAT shows various undo and transaction statistics, such as the amount of undo space consumed in the instance.

Undo Mode

Undo mode provides a more flexible way to migrate from manual undo management to automatic undo management. A database system can run in either manual undo management mode or automatic undo management mode. In manual undo management mode, undo space is managed through rollback segments.

Manual undo management mode is supported under any compatibility level. Use it when you need to run Oracle9i to take advantage of some new features, but are not yet not ready to convert to automatic undo management mode.

In automatic undo management mode, undo space is managed in undo tablespaces. To use automatic undo management mode, the database administrator needs only to create an undo tablespace for each instance and set the UNDO_MANAGEMENT initialization parameter to AUTO. Automatic undo management mode is supported under compatibility levels of Oracle9i or higher. Although manual undo management mode is supported, you are strongly encouraged to run in automatic undo management mode.

3.2. Tablespaces, Datafiles, and Control Files

Introduction to Tablespaces, Datafiles, and Control Files

Oracle stores data logically in tablespaces and physically in datafiles associated with the corresponding tablespace. Figure 3 illustrates this relationship.

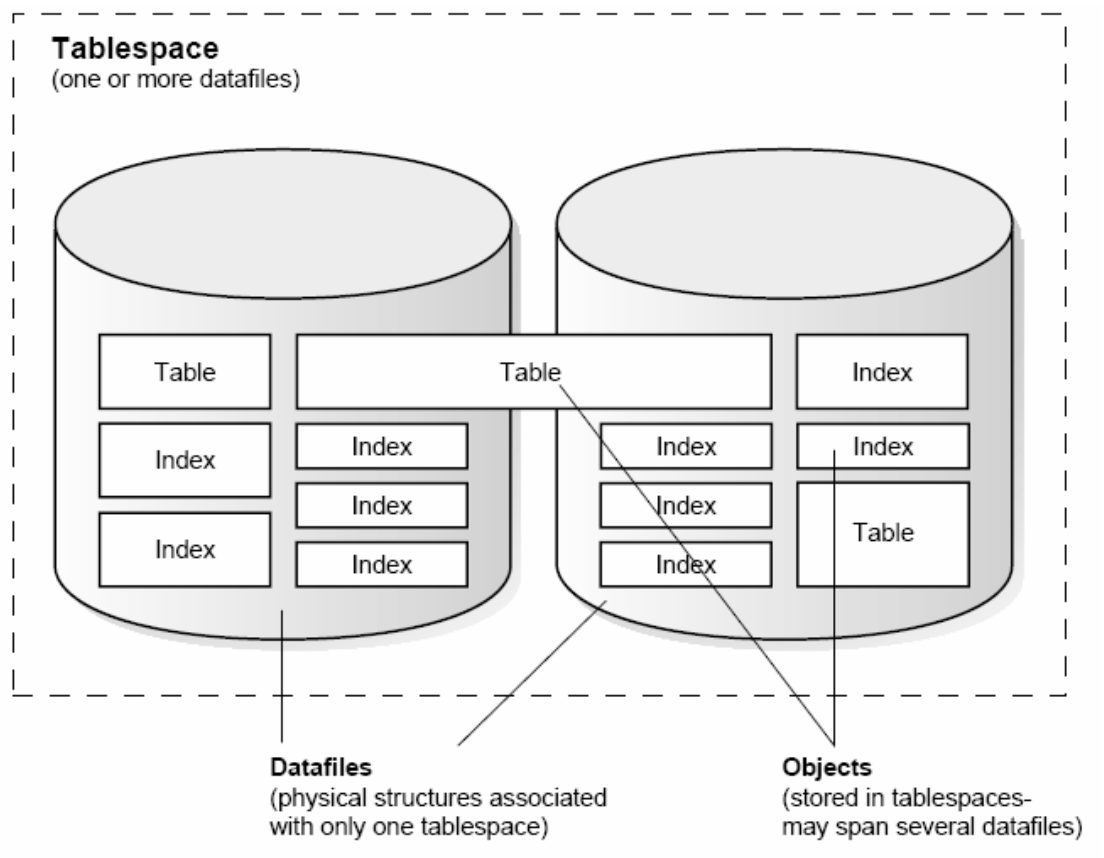

Figure 3. Datafiles and Tablespaces.

Databases, tablespaces, and datafiles are closely related, but they have important differences:

- An Oracle database consists of one or more logical storage units called tablespaces, which collectively store all of the database's data.

- Each tablespace in an Oracle database consists of one or more files called datafiles, which are physical structures that conform to the operating system in which Oracle is running.
- A database's data is collectively stored in the datafiles that constitute each tablespace of the database. For example, the simplest Oracle database would have one tablespace and one datafile. Another database can have three tablespaces, each consisting of two datafiles (for a total of six datafiles).

Oracle-Managed Files

Oracle-managed files eliminate the need for you, the DBA, to directly manage the operating system files comprising an Oracle database. You specify operations in terms of database objects rather than filenames. Oracle internally uses standard file system interfaces to create and delete files as needed for the following database structures:

- Tablespaces
- Online redo log files
- Control files

Through initialization parameters, you specify the file system directory to be used for a particular type of file. Oracle then ensures that a unique file, an Oracle-managed file, is created and deleted when no longer needed.

Allocate More Space for a Database

The size of a tablespace is the size of the datafiles that constitute the tablespace. The size of a database is the collective size of the tablespaces that constitute the database.

You can enlarge a database in three ways:

- Add a datafile to a tablespace
- Add a new tablespace
- Increase the size of a datafile

When you add another datafile to an existing tablespace, you increase the amount of disk space allocated for the corresponding tablespace. Figure 4 illustrates this kind of space increase.

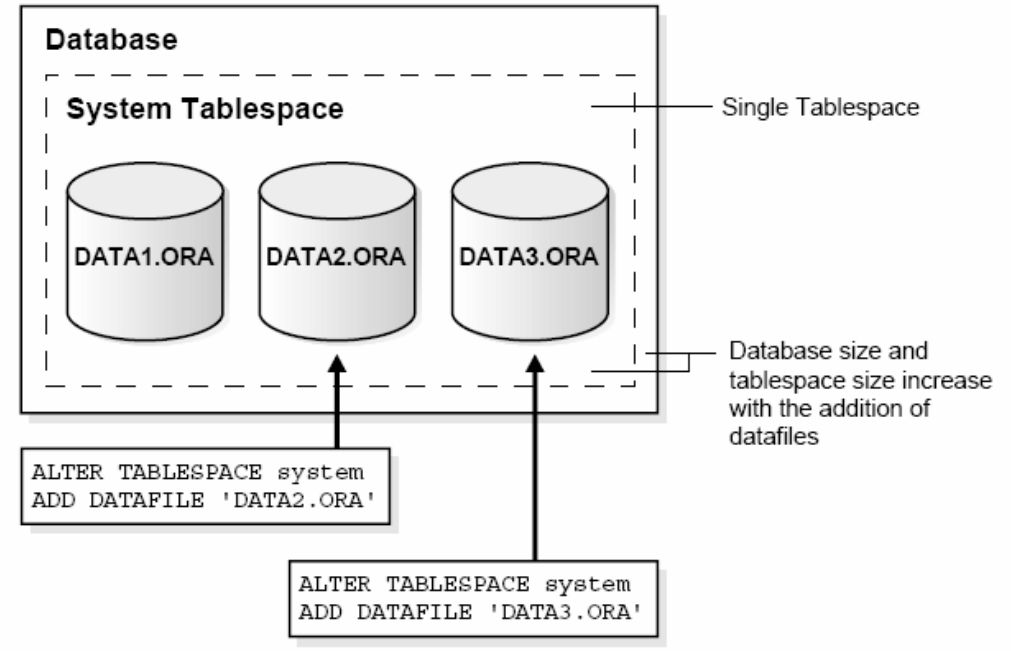

Fig. 4. Enlarging a Database by Adding a Datafile to a Tablespace.

Alternatively, you can create a new tablespace (which contains at least one additional datafile) to increase the size of a database. Figure 5 illustrates this.

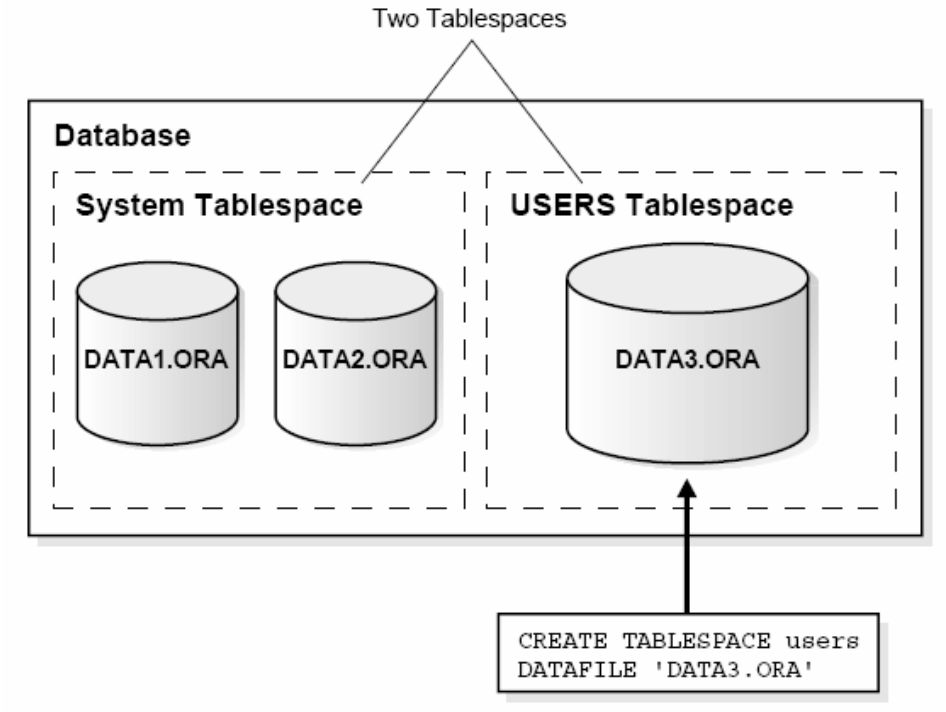

Fig. 5. Enlarging a Database by Adding a New Tablespace.

The third option for enlarging a database is to change a datafile's size or let datafiles in existing tablespaces grow dynamically as more space is needed. You accomplish this by altering existing files or by adding files with dynamic extension properties. Figure 6 illustrates this.

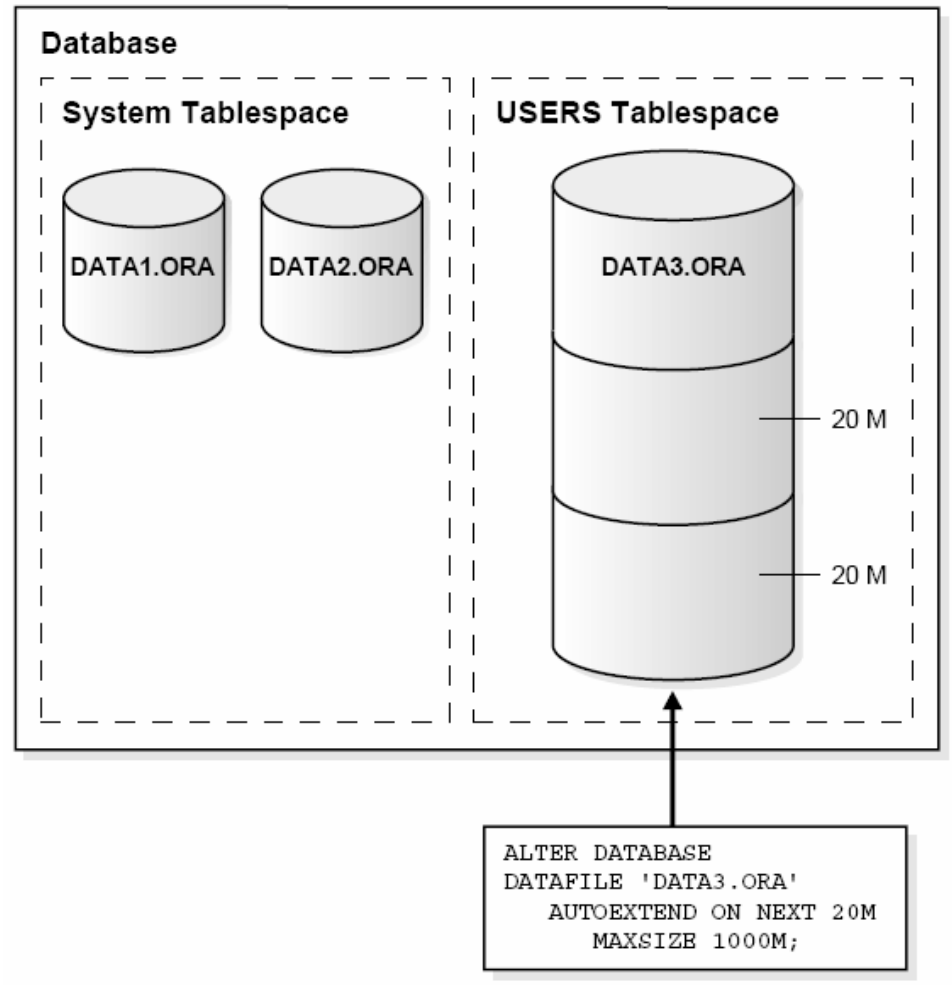

Fig. 6. Enlarging a Database by Dynamically Sizing Datafiles.

Tablespace Overview

A database is divided into one or more logical storage units called **tablespaces**. Tablespaces are divided into logical units of storage called **segments**, which are further divided into **extents**. Extents are a collection of contiguous **blocks**.

The SYSTEM Tablespace

Every Oracle database contains a tablespace named SYSTEM, which Oracle creates automatically when the database is created. The SYSTEM tablespace is always online when the database is open.

To take advantage of the benefits of locally managed tablespaces, you can create a locally managed SYSTEM tablespace, or you can migrate an existing dictionary managed SYSTEM tablespace to a locally managed format.

In a database with a locally managed SYSTEM tablespace, dictionary tablespaces cannot be created. It is possible to plug in a dictionary managed tablespace using the transportable feature, but it cannot be made writable.

NOTE:

Once a tablespace is locally managed, it cannot be reverted back to being dictionary managed.

The Data Dictionary

The SYSTEM tablespace always contains the data dictionary tables for the entire database. The data dictionary tables are stored in *datafile 1*.

PL/SQL Program Units Description

All data stored on behalf of stored PL/SQL program units (that is, procedures, functions, packages, and triggers) resides in the SYSTEM tablespace. If the database contains many of these program units, then the database administrator must provide the space the units need in the SYSTEM tablespace.

Undo Tablespaces

Undo tablespaces are special tablespaces used solely for storing undo information. You cannot create any other segment types (for example, tables or indexes) in undo tablespaces. Each database contains zero or more undo tablespaces. In automatic undo management mode, each Oracle instance is assigned one (and only one) undo tablespace. Undo data is managed within an undo tablespace using undo segments that are automatically created and maintained by Oracle.

When the first DML operation is run within a transaction, the transaction is bound (assigned) to an undo segment (and therefore to a transaction table) in the current undo tablespace. In rare circumstances, if the instance does not have a designated undo tablespace, the transaction binds to the system undo segment.

Each undo tablespace is composed of a set of undo files and is locally managed. Like other types of tablespaces, undo blocks are grouped in extents and the status of each extent is represented in the bitmap. At any point in time, an extent is either allocated to (and used by) a transaction table, or it is free.

Creation of Undo Tablespaces

A database administrator creates undo tablespaces individually, using the CREATE UNDO TABLESPACE statement. It can also be created when the database is created, using the CREATE DATABASE statement. A set of files is assigned to each newly created undo tablespace. Like regular tablespaces, attributes of undo tablespaces can be modified with the ALTER TABLESPACE statement and dropped with the DROP TABLESPACE statement.

NOTE:

An undo tablespace cannot be dropped if it is being used by any instance or contains any undo

information needed to recover transactions.

Assignment of Undo Tablespaces

You assign an undo tablespace to an instance in one of two ways:

- At instance startup. You can specify the undo tablespace in the initialization file or let the system choose an available undo tablespace.
- While the instance is running. Use ALTER SYSTEM SET UNDO_TABLESPACE to replace the active undo tablespace with another undo tablespace. This method is rarely used.

You can add more space to an undo tablespace by adding more data files to the undo tablespace with the ALTER TABLESPACE statement.

You can have more than one undo tablespace and switch between them. Use the Database Resource Manager to establish user quotas for undo tablespaces. You can specify the retention period for undo information.

Default Temporary Tablespace

When the SYSTEM tablespace is locally managed, you must define a default temporary tablespace when creating a database. A locally managed SYSTEM tablespace cannot be used for default temporary storage.

If SYSTEM is dictionary managed and if you do not define a default temporary tablespace when creating the database, then SYSTEM is still used for default temporary storage. However, you will receive a warning in ALERT.LOG saying that a default temporary tablespace is recommended and will be necessary in future releases.

How to Specify a Default Temporary Tablespace

Specify a default temporary tablespace when you create a database, using the DEFAULT TEMPORARY TABLESPACE extension to the CREATE DATABASE statement.

If you drop the default temporary tablespace, then the SYSTEM tablespace is used as the default temporary tablespace.

NOTE:

You cannot make the default temporary tablespace permanent or take it offline.

Using Multiple Tablespaces

A very small database may need only the SYSTEM tablespace; however, Oracle Corporation recommends that you create at least one additional tablespace to store user data separate from data dictionary information. This gives you more flexibility in various database administration operations and reduces contention among dictionary objects and schema objects for the same datafiles.

You can use multiple tablespaces to perform the following tasks:

- Control disk space allocation for database data

- Assign specific space quotas for database users
- Control availability of data by taking individual tablespaces online or offline
- Perform partial database backup or recovery operations
- Allocate data storage across devices to improve performance

A database administrator can use tablespaces to do the following actions:

- Create new tablespaces
- Add datafiles to tablespaces
- Set and alter default segment storage settings for segments created in a tablespace
- Make a tablespace read-only or read/write
- Make a tablespace temporary or permanent
- Drop tablespaces.

Managing Space in Tablespaces

Tablespaces allocate space in extents. Tablespaces can use two different methods to keep track of their free and used space:

- Locally managed tablespaces: Extent management by the tablespace
- Dictionary managed tablespaces: Extent management by the data dictionary

When you create a tablespace, you choose one of these methods of space management. You cannot alter the method at a later time.

Locally Managed Tablespaces

A tablespace that manages its own extents maintains a bitmap in each datafile to keep track of the free or used status of blocks in that datafile. 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 (except for special cases such as tablespace quota information).

Locally managed tablespaces have the following advantages over dictionary managed tablespaces:

- Local management of extents automatically tracks adjacent free space, eliminating the need to coalesce free extents.
- Local management of extents avoids recursive space management operations. Such recursive operations can occur in dictionary managed tablespaces if consuming or releasing space in an extent results in another operation that consumes or releases space in a data dictionary table or rollback segment.

The sizes of extents that are managed locally can be determined automatically by the system. Alternatively, all extents can have the same size in a locally managed tablespace and override object storage options.

The LOCAL clause of the CREATE TABLESPACE or CREATE TEMPORARY TABLESPACE statement is specified to create locally managed permanent or temporary tablespaces, respectively.

Segment Space Management in Locally Managed Tablespaces

When you create a locally managed tablespace using the CREATE TABLESPACE statement, the SEGMENT SPACE MANAGEMENT clause lets you specify how free and used space within a segment is to be managed. Your choices are:

- AUTO

  This keyword tells Oracle that you want to use bitmaps to manage the free space within segments. A bitmap, in this case, is a map that describes the status of each data block within a segment with respect to the amount of space in the block available for inserting rows. As more or less space becomes available in a data block, its new state is reflected in the bitmap. Bitmaps enable Oracle to manage free space more automatically; thus, this form of space management is called automatic segment-space management.

- MANUAL

  This keyword tells Oracle that you want to use free lists for managing free space within segments. Free lists are lists of data blocks that have space available for inserting rows. MANUAL is the default.

Dictionary Managed Tablespaces

If you created your database with an earlier version of Oracle, then you could be using dictionary managed tablespaces. For a tablespace that uses the data dictionary to manage its extents, Oracle updates the appropriate tables in the data dictionary whenever an extent is allocated or freed for reuse. Oracle also stores rollback information about each update of the dictionary tables. Because dictionary tables and rollback segments are part of the database, the space that they occupy is subject to the same space management operations as all other data.

Multiple Block Sizes

The block size of the SYSTEM tablespace is the standard block size. This is set when the database is created and can be any valid size. You can specify up to four block sizes, in addition to a standard block size. In the initialization file, you can configure subcaches within the buffer cache for each of these block sizes. Subcaches can also be configured while an instance is running.

You can create tablespaces having any of these block sizes. The standard block size is used for the system tablespace and most other tablespaces.

NOTE:

All partitions of a partitioned object must reside in tablespaces of a single block size.

Multiple block sizes are useful primarily when transporting a tablespace from an OLTP database to an enterprise data warehouse. This facilitates transport between databases of different block sizes.

Online and Offline Tablespaces

A database administrator can bring any tablespace other than the SYSTEM tablespace **online** (accessible) or **offline** (not accessible) whenever the database is open. The SYSTEM tablespace is always online when the database is open because the data dictionary must always be available to Oracle.

A tablespace is usually online so that the data contained within it is available to database users. However, the database administrator can take a tablespace offline for maintenance or backup and recovery purposes.

When a Tablespace Goes Offline

When a tablespace goes offline, Oracle does not permit any subsequent SQL statements to reference objects contained in that tablespace. Active transactions with completed statements that refer to data in that tablespace are not affected at the transaction level. Oracle saves rollback data corresponding to those completed statements in a deferred rollback segment in the SYSTEM tablespace. When the tablespace is brought back online, Oracle applies the rollback data to the tablespace, if needed.

When a tablespace goes offline or comes back online, this is recorded in the data dictionary in the SYSTEM tablespace. If a tablespace is offline when you shut down a database, the tablespace remains offline when the database is subsequently mounted and reopened.

You can bring a tablespace online only in the database in which it was created because the necessary data dictionary information is maintained in the SYSTEM tablespace of that database. An offline tablespace cannot be read or edited by any utility other than Oracle. Thus, offline tablespaces cannot be transposed to other databases.

Oracle automatically switches a tablespace from online to offline when certain errors are encountered. For example, Oracle switches a tablespace from online to offline when the database writer process, DBWn, fails in several attempts to write to a datafile of the tablespace. Users trying to access tables in the offline tablespace receive an error. If the problem that causes this disk I/O to fail is media failure, you must recover the tablespace after you correct the problem.

Use of Tablespaces for Special Procedures

If you create multiple tablespaces to separate different types of data, you take specific tablespaces offline for various procedures. Other tablespaces remain online, and the information in them is still available for use. However, special circumstances can occur when tablespaces are taken offline. For example, if two tablespaces are used to separate table data from index data, the following is true:

- If the tablespace containing the indexes is offline, then queries can still access table data because queries do not require an index to access the table data.
- If the tablespace containing the tables is offline, then the table data in the database is not accessible because the tables are required to access the data.

If Oracle has enough information in the online tablespaces to run a statement, it does so. If it needs data in an offline tablespace, then it causes the statement to fail.

Read-Only Tablespaces

The primary purpose of read-only tablespaces is to eliminate the need to perform backup and recovery of large, static portions of a database. Oracle never updates the files of a read-only tablespace, and therefore the files can reside on read-only media such as CD-ROMs or WORM drives.

Read-only tablespaces cannot be modified. To update a read-only tablespace, first make the tablespace read/write. After updating the tablespace, you can then reset it to be read-only.

Because read-only tablespaces cannot be modified, and as long as they have not been made read-write at any point, they do not need repeated backup. Also, if you need to recover your database, you do not need to recover any read-only tablespaces, because they could not have been modified.

Temporary Tablespaces for Sort Operations

You can manage space for sort operations more efficiently by designating temporary tablespaces exclusively for sorts. Doing so effectively eliminates serialization of space management operations involved in the allocation and deallocation of sort space.

All operations that use sorts, including joins, index builds, ordering, computing aggregates (GROUP BY), and collecting optimizer statistics, benefit from temporary tablespaces. The performance gains are significant with Real Application Clusters.

Sort Segments

A temporary tablespace can be used only for sort segments. A temporary tablespace is not the same as a tablespace that a user designates for temporary segments, which can be any tablespace available to the user. No permanent schema objects can reside in a temporary tablespace.

Sort segments are used when a segment is shared by multiple sort operations. One sort segment exists for every instance that performs a sort operation in a given tablespace.

Temporary tablespaces provide performance improvements when you have multiple sorts that are too large to fit into memory. The sort segment of a given temporary tablespace is created at the time of the first sort operation. The sort segment expands by allocating extents until the segment size is equal to or greater than the total storage demands of all of the active sorts running on that instance.

Creation of Temporary Tablespaces

You can create temporary tablespaces by using the CREATE TABLESPACE or CREATE TEMPORARY TABLESPACE statement.

Transport of Tablespaces Between Databases

A transportable tablespace lets you move a subset of an Oracle database from one Oracle database to another on the same platform. You can clone a tablespace and plug it into another database, copying the tablespace between databases, or you can unplug a tablespace from one Oracle database and plug it into another Oracle database, moving the tablespace between databases on the same platform.

Moving data by transporting tablespaces can be orders of magnitude faster than either export/import or unload/load of the same data, because transporting a tablespace involves only copying datafiles and integrating the tablespace metadata. When you transport tablespaces you can also move index data, so you do not have to rebuild the indexes after importing or loading the table data.

NOTE:

You can transport tablespaces only between Oracle databases that use the same character set and that run on compatible platforms from the same hardware vendor.

How to Move or Copy a Tablespace to Another Database

To move or copy a set of tablespaces, you must make the tablespaces read-only, copy the datafiles of these tablespaces, and use export/import to move the database information (metadata) stored in the data dictionary. Both the datafiles and the metadata export file must be copied to the target database. The transport of these files can be done using any facility for copying flat files, such as the operating system copying facility, ftp, or publishing on CDs.

After copying the datafiles and importing the metadata, you can optionally put the tablespaces in read/write mode.

NOTE:

In a database with a locally managed SYSTEM tablespace, dictionary tablespaces cannot be created. It is possible to plug in a dictionary managed tablespace using the transportable feature, but it cannot be made writable.

Datafiles Overview

Atablespace in an Oracle database consists of one ormore physical datafiles. A datafile can be associated with only one tablespace and only one database.

Oracle creates a datafile for a tablespace by allocating the specified amount of disk space plus the overhead required for the file header. When a datafile is created, the operating system under which Oracle runs is responsible for clearing old information and authorizations from a file before allocating it to Oracle. If the file is large, this process can take a significant amount of time. The first tablespace in any database is always the SYSTEM tablespace, so Oracle automatically allocates the first datafiles of any database for the SYSTEM tablespace during database creation.

Datafile Contents

When a datafile is first created, the allocated disk space is formatted but does not contain any user data. However, Oracle reserves the space to hold the data for future segments of the associated tablespace—it is used exclusively by Oracle. As the data grows in a tablespace, Oracle uses the free space in the associated datafiles to allocate extents for the segment.

The data associated with schema objects in a tablespace is physically stored in one or more of the datafiles that constitute the tablespace. Note that a schema object does not correspond to a specific datafile; rather, a datafile is a repository for the data of any schema object within a specific tablespace. Oracle allocates space for the data associated with a schema object in one or more datafiles of a tablespace. Therefore, a schema object can span one or more datafiles. Unless table striping is used (where data is spread across more than one disk), the database administrator and end users cannot control which datafile stores a schema object.

Size of Datafiles

You can alter the size of a datafile after its creation or you can specify that a datafile should dynamically grow as schema objects in the tablespace grow. This functionality enables you to have fewer datafiles for each tablespace and can simplify administration of datafiles.

NOTE:

You need sufficient space on the operating system for expansion.

Offline Datafiles

You can take tablespaces offline or bring them online at any time, except for the SYSTEM tablespace. All of the datafiles of a tablespace are taken offline or brought online as a unit when you take the tablespace offline or bring it online, respectively.

You can take individual datafiles offline. However, this is usually done only during some database recovery procedures.

Temporary Datafiles

Locally managed temporary tablespaces have temporary datafiles (tempfiles), which are similar to ordinary datafiles, with the following exceptions:

- Tempfiles are always set to NOLOGGING mode.
- You cannot make a tempfile read-only.
- You cannot rename a tempfile.
- You cannot create a tempfile with the ALTER DATABASE statement.
- When you create or resize tempfiles, they are not always guaranteed allocation of disk space for the file size specified. On certain file systems (for example, UNIX) disk blocks are allocated not at file creation or resizing, but before the blocks are accessed.
- Tempfile information is shown in the dictionary view DBA_TEMP_FILES and the dynamic performance view V$TEMPFILE, but not in DBA_DATA_FILES or the V$DATAFILE view.

Control Files Overview

The database control file is a small binary file necessary for the database to start and operate successfully. A control file is updated continuously by Oracle during database use, so it must be available for writing whenever the database is open. If for some reason the control file is not accessible, then the database cannot function properly.

Each control file is associated with only one Oracle database.

Control File Contents

A control file contains information about the associated database that is required for access by an instance, both at startup and during normal operation. Control file information can be modified only by Oracle; no database administrator or user can edit a control file.

Among other things, a control file contains information such as:

- The database name
- The timestamp of database creation
- The names and locations of associated datafiles and online redo log files

- Tablespace information
- Datafile offline ranges
- The log history
- Archived log information
- Backup set and backup piece information
- Backup datafile and redo log information
- Datafile copy information
- The current log sequence number
- Checkpoint information

The database name and timestamp originate at database creation. The database name is taken from either the name specified by the initialization parameter DB_NAME or the name used in the CREATE DATABASE statement.

Each time that a datafile or an online redo log file is added to, renamed in, or dropped from the database, the control file is updated to reflect this physical structure change. These changes are recorded so that:

- Oracle can identify the datafiles and online redo log files to open during database startup
- Oracle can identify files that are required or available in case database recovery is necessary

Therefore, if you make a change to the physical structure of your database (using ALTER DATABASE statements), then you should immediately make a backup of your control file.

Control files also record information about checkpoints. Every three seconds, the checkpoint process (CKPT) records information in the control file about the checkpoint position in the online redo log. This information is used during database recovery to tell Oracle that all redo entries recorded before this point in the online redo log group are not necessary for database recovery; they were already written to the datafiles.

Multiplexed Control Files

As with online redo log files, Oracle enables multiple, identical control files to be open concurrently and written for the same database.

By storing multiple control files for a single database on different disks, you can safeguard against a single point of failure with respect to control files. If a single disk that contained a control file crashes, then the current instance fails when Oracle attempts to access the damaged control file. However, when other copies of the current control file are available on different disks, an instance can be restarted easily without the need for database recovery.

If all control files of a database are permanently lost during operation, then the instance is aborted and media recovery is required. Media recovery is not straightforward if an older backup of a control file must be used because a current copy is not available. Therefore, it is strongly recommended that you adhere to the following practices:

- Use multiplexed control files with each database
- Store each copy on a different physical disk
- Use operating system mirroring

- Monitor backups

3.3. The Data Dictionary

One of the most important parts of an Oracle database is its data dictionary, which is a read-only set of tables that provides information about the database. A data dictionary contains:

- The definitions of all schema objects in the database (tables, views, indexes, clusters, synonyms, sequences, procedures, functions, packages, triggers, and so on)
- How much space has been allocated for, and is currently used by, the schema objects
- Default values for columns
- Integrity constraint information
- The names of Oracle users
- Privileges and roles each user has been granted
- Auditing information, such as who has accessed or updated various schema objects
- Other general database information

The data dictionary is structured in tables and views, just like other database data. All the data dictionary tables and views for a given database are stored in that database's SYSTEM tablespace.

Not only is the data dictionary central to every Oracle database, it is an important tool for all users, from end users to application designers and database administrators. Use SQL statements to access the data dictionary. Because the data dictionary is read-only, you can issue only queries (SELECT statements) against it's tables and views.

Structure of the Data Dictionary

The data dictionary consists of the following:

Base Tables

The underlying tables that store information about the associated database. Only Oracle should write to and read these tables. Users rarely access them directly because they are normalized, andmost of the data is stored in a cryptic format.

User-Accessible Views

The views that summarize and display the information stored in the base tables of the data dictionary. These views decode the base table data into useful information, such as user or table names, using joins and WHERE clauses to simplify the information. Most users are given access to the views rather than the base tables.

SYS, Owner of the Data Dictionary

The Oracle user SYS owns all base tables and user-accessible views of the data dictionary. No Oracle user should ever alter (UPDATE, DELETE, or INSERT) any rows or schema objects contained in the SYS schema, because such activity can compromise data integrity. The security administrator must keep strict control of this central account.

CAUTION:

Altering or manipulating the data in data dictionary tables can permanently and detrimentally affect the operation of a database.

How the Data Dictionary Is Used

The data dictionary has three primary uses:

- Oracle accesses the data dictionary to find information about users, schema objects, and storage structures.
- Oracle modifies the data dictionary every time that a data definition language (DDL) statement is issued.
- Any Oracle user can use the data dictionary as a read-only reference for information about the database.

How Oracle Uses the Data Dictionary

Data in the base tables of the data dictionary is necessary for Oracle to function. Therefore, only Oracle should write or change data dictionary information. Oracle provides scripts to modify the data dictionary tables when a database is upgraded or downgraded.

During database operation, Oracle reads the data dictionary to ascertain that schema objects exist and that users have proper access to them. Oracle also updates the data dictionary continuously to reflect changes in database structures, auditing, grants, and data.

For example, if user Kathy creates a table named parts, then new rows are added to the data dictionary that reflect the new table, columns, segment, extents, and the privileges that Kathy has on the table. This new information is then visible the next time the dictionary views are queried.

Public Synonyms for Data Dictionary Views

Oracle creates public synonyms for many data dictionary views so users can access them conveniently. The security administrator can also create additional public synonyms for schema objects that are used system-wide. Users should avoid naming their own schema objects with the same names as those used for public synonyms.

Cache the Data Dictionary for Fast Access

Much of the data dictionary information is kept in the SGA in the dictionary cache, because Oracle constantly accesses the data dictionary during database operation to validate user access and to verify the state of schema objects. All information is stored in memory using the least recently used (LRU) algorithm.

Parsing information is typically kept in the caches. The COMMENTS columns describing the tables and their columns are not cached unless they are accessed frequently.

Other Programs and the Data Dictionary

Other Oracle products can reference existing views and create additional data dictionary tables or views of their own. Application developers who write programs that refer to the data dictionary should refer to the public synonyms rather than the underlying tables: the synonyms are less likely to change between software releases.

How to Use the Data Dictionary

The views of the data dictionary serve as a reference for all database users. Access the data dictionary views with SQL statements. Some views are accessible to all Oracle users, and others are intended for database administrators only.

The data dictionary is always available when the database is open. It resides in the SYSTEM tablespace, which is always online.

The data dictionary consists of sets of views. In many cases, a set consists of three views containing similar information and distinguished from each other by their prefixes:

Table 1. Data Dictionary View Prefixes.

Prefix	Scope
USER	User's view (what is in the user's schema)
ALL	Expanded user's view (what the user can access)
DBA	Database administrator's view (what is in all users' schemas)

The set of columns is identical across views, with these exceptions:

Views with the prefix USER usually exclude the column OWNER. This column is implied in the USER views to be the user issuing the query.

Some DBA views have additional columns containing information useful to the administrator.

Views with the Prefix USER

The views most likely to be of interest to typical database users are those with the prefix USER. These views:

Refer to the user's own private environment in the database, including information about schema objects created by the user, grants made by the user, and so on

Display only rows pertinent to the user

Have columns identical to the other views, except that the column OWNER is implied

Return a subset of the information in the ALL views

Can have abbreviated PUBLIC synonyms for convenience

For example, the following query returns all the objects contained in your schema:

```
SELECT object_name, object_type FROM USER_OBJECTS;
```

Views with the Prefix ALL

Views with the prefix ALL refer to the user's overall perspective of the database. These views return information about schema objects to which the user has access through public or explicit grants of privileges and roles, in addition to schema objects that the user owns. For example, the following query returns information about all the objects to which you have access:

```
SELECT owner, object_name, object_type FROM ALL_OBJECTS;
```

Views with the Prefix DBA

Views with the prefix DBA show a global view of the entire database. Synonyms are not created for these views, because DBA views should be queried only by administrators. Therefore, to query the DBA views, administrators must prefix the view name with its owner, SYS, as in the following:

```
SELECT owner, object_name, object_type FROM SYS.DBA_OBJECTS;
```

Oracle recommends that you implement data dictionary protection to prevent users having the ANY system privileges from using such privileges on the data dictionary. If you enable dictionary protection (O7_DICTIONARY_ACCESSIBILITY is false), then access to objects in the SYS schema (dictionary objects) is restricted to users with the SYS schema. These users are SYS and those who connect as SYSDBA.

Dynamic Performance Tables

Throughout its operation, Oracle maintains a set of virtual tables that record current database activity. These tables are called dynamic performance tables.

Dynamic performance tables are not true tables, and they should not be accessed by most users. However, database administrators can query and create views on the tables and grant access to those views to other users. These views are sometimes called fixed views because they cannot be altered or removed by the database administrator.

SYS owns the dynamic performance tables; their names all begin with V_$. Views are created on these tables, and then public synonyms are created for the views. The synonym names begin with V$. For example, the V$DATAFILE view contains information about the database's datafiles, and the V$FIXED_TABLE view contains information about all of the dynamic performance tables and views in the database.

Database Object Metadata

The DBMS_METADATA package provides interfaces for extracting complete definitions of database objects. The definitions can be expressed either as XML or as SQL DDL. Two styles of interface are provided:

- A flexible, sophisticated interface for programmatic control
- A simplified interface for ad hoc querying

References

[1] Bob Bryla, "*Oracle 9i DBA Jump Start*", Sybex, 2003.

[2] Michael J. Hernandez, "Proiectarea Bazelor de Date", Teora, 2003.

[3] Oracle9i Database Concepts, Oracle, 2002.