

Chapter 8

Introducing Oracle Database 12c Components and Architecture

ORACLE DATABASE 12c: OCA EXAM OBJECTIVES COVERED IN THIS CHAPTER:

✓ Exploring the Oracle Database Architecture

- List the architectural components of Oracle Database.
- Explain the memory structures.
- Describe the background processes.
- Explain the relationship between logical and physical storage structures.

✓ Oracle Database Management Tools

- Use database management tools.





With this chapter, you'll start learning Oracle Database 12c (Oracle 12c) database administration. This chapter and the remaining chapters of the book will discuss the objectives for the *Oracle Database 12c: Installation and Administration* OCA certification exam. There are two parts to the exam: *Oracle Database Administration* and *Installing, Upgrading, and Patching the Oracle Database*. The book's chapters are organized to give you a natural progression from the basics, installation, database creation, basic administration, and on to advanced topics, not necessarily organized in the order of examination objectives specified by Oracle.

With the release of Oracle Database 12c, the Oracle Corporation has delivered a powerful and feature-rich database that can meet the performance, availability, recoverability, multitenancy, cloud-enabled, application-testing, and security requirements of any mission-critical application. As the Oracle DBA, you are responsible for managing and maintaining the Oracle Database 12c environment throughout its lifecycle, from the initial installation through creation, configuration, final deployment, and its daily administration. Performing these tasks requires a solid understanding of Oracle's product offerings so that you can apply the proper tools and features to the application. You must also use relational database concepts to design, implement, and maintain the tables that store the application data. At the heart of these activities is the need for a thorough understanding of the Oracle architecture and the tools and techniques used to monitor and manage the components of this architecture.

We will begin this chapter by reviewing the Oracle database basics. You will learn what constitutes the Oracle Database 12c architecture. We'll provide an overview of the memory structures, the processes that manage the database, how data is stored in the database, and the many pluggable databases in a consolidated cloud database. We will also discuss the tools used to administer Oracle Database 12c.

In this chapter, you will see many examples of using the database to show information from the database, and you might wonder, "Why are you doing this without first showing me how to create a database?" It is a chicken-and-egg situation; we think knowing the basic components and high-level architecture will help you better understand the database and create options. So, you will learn the architecture basics in this chapter and actually create a database in the next chapter.



Exam objectives are subject to change at any time without prior notice and at Oracle's sole discretion. Please visit Oracle's Training and Certification website at <http://education.oracle.com> for the most current exam-objectives listing.

Oracle Database Fundamentals

Databases store data. The data itself is composed of related logical units of information. The *database management system* (DBMS) facilitates the storage, modification, and retrieval of this data. Some early database technologies used flat files or hierarchical file structures to store application data. Others used networks of connections between sets of data to store and locate information. The early DBMS architecture mixed the physical manipulation of data with its logical manipulation. When the location of data changed, the application referencing the data had to be updated. Relational databases brought a revolutionary change to this architecture. Relational DBMS introduced data independence, which separated the physical model of the data from its logical model. Oracle is a relational DBMS.

All releases of Oracle's database products have used a relational DBMS model to store data in the database. This relational model is based on the groundbreaking work of Dr. Edgar Codd, which was first published in 1970 in his paper "A Relational Model of Data for Large Shared Data Banks." IBM Corporation, which was then an early adopter of Dr. Codd's model, helped develop the computer language that is used to access all relational databases today—Structured Query Language (SQL). The great thing about SQL is that you can use it to easily interact with relational databases without having to write complex computer programs and without needing to know where or how the data is physically stored on disk. You saw several SQL statements in the previous chapters.

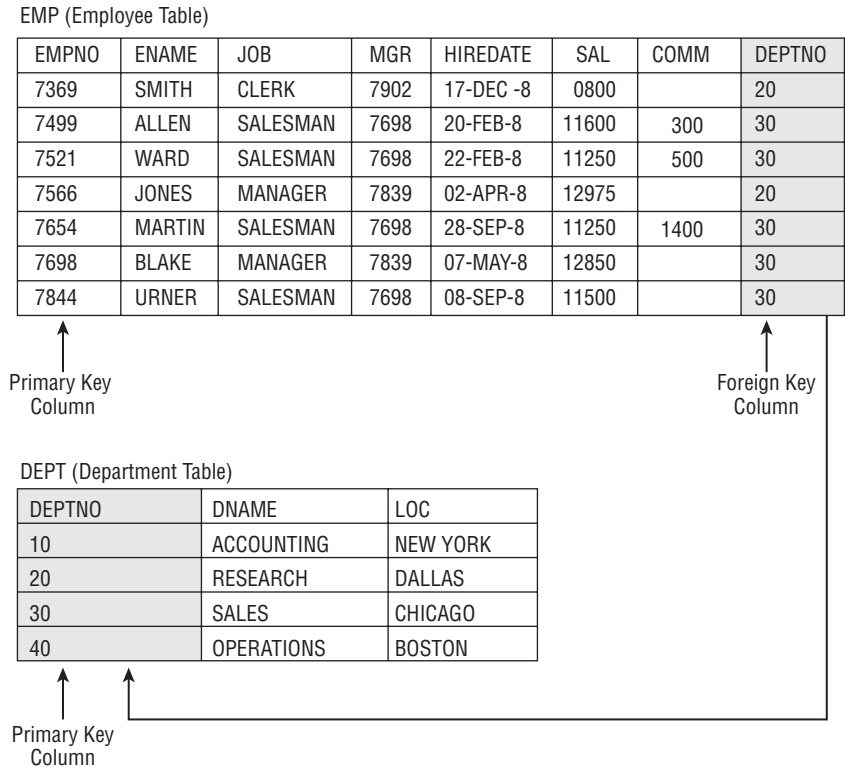
Relational Databases

The concept of a *relational database management system* (RDBMS) is that the data consists of a set of relational objects. The basic storage of data in a database is a table. The relations are implemented in tables, where data is stored in rows and columns. Figure 8.1 shows such a relationship.

The DEPT table in the lower part of the figure stores information about departments in the company. Each department is identified by the department ID. Along with the ID, the name and location of the department are also stored in the table. The EMP table stores information about the employees in the company. Each employee is identified by a unique employee ID. This table includes employee information such as hire date, salary, manager, and so on. The DEPTNO column in both tables provides a relationship between the tables. A department may have many employees, but an employee can work for only one department.

Because the user accessing this data doesn't need to know how or where the row is stored in the database, there must be a way to uniquely identify the rows in the tables. In our example, the department is uniquely identified by department number, and an employee is identified by an employee ID. The column (or set of columns) that uniquely identifies a row is known as the *primary key*. According to relational theory, each table in a relational database must have a primary key.

FIGURE 8.1 Relational tables



When relating tables together, the primary key of one table is placed in another table. For example, the primary key of the DEPT table is a column in the EMP table. In RDBMS terminology, this is known as a *foreign key*. A foreign key states that the data value in the column exists in another table and should continue to exist in the other table to keep the relationship between tables. The table where the column is a primary key is known as the *parent table*, and the table where the foreign key column exists is known as the *child table*. Oracle enforces the parent-child relationship between tables using *constraints*.

Oracle Database 12c Objects

Every RDBMS supports a variety of database objects. Oracle Database 12c supports the entire set of database objects required for a relational and object-relational database, such as tables, views, constraints, and so on. It also supports a wide range of objects specific to the Oracle database, such as packages, sequences, materialized views, and so on. Table 8.1 lists the major commonly used objects in Oracle Database 12c.

TABLE 8.1 Oracle Database 12c Objects

Object Type	Description
Table	A table is the basic form of data storage. A table has columns and stores rows of data.
View	A view is a stored query. No data-storage space is occupied for view data.
Index	An index is an optional structure that is useful for fetching data faster.
Materialized view	Materialized views are used to summarize and store data. They are similar to views but take up storage space to store data.
Index-organized table	An index-organized table stores the table data along with the index, instead of storing table and index separately.
Cluster	A cluster is a group of tables sharing a common column. The cluster stores the rows of the tables together with the common columns stored once.
Constraint	A constraint is a stored rule to enforce data integrity.
Sequence	A sequence provides a mechanism for the continuous generation of numbers.
Synonym	A synonym is an alias for a database schema object.
Trigger	A trigger is a PL/SQL program unit that is executed when an event occurs.
Stored function	Stored functions are PL/SQL programs that can be used to create user-defined functions to return a value.
Stored procedure	Stored procedures are PL/SQL programs to define a business process.
Package	A package is a collection of procedures, functions, and other program constructs.
Java	Stored Java procedures can be created in Oracle to define business processes.
Database link	Database links are used to communicate between databases to share data.

You use SQL to create database objects and to interact with application data. In the next section, we will discuss the tools available to access and administer Oracle Database 12c.

Interacting with Oracle Database 12c

Several Oracle database management tools are available for the DBA to interact with and manage Oracle Database 12c. SQL is the language used to interact with Oracle Database 12c. The common tools available for the DBA to administer Oracle Database 12c are as follows:

- *SQL*Plus*, which is a SQL command-line interface utility
- *SQL Developer*, a GUI tool to explore and manage the database using predefined menu actions and SQL statements
- *Oracle Enterprise Manager Database Express 12c*, a GUI tool for database administration and performance management

Using SQL*Plus and SQL Developer, you interact directly with Oracle Database 12c using SQL statements and a superset of commands such as STARTUP, SHUTDOWN, and so on. Using Enterprise Manager, you interact indirectly with Oracle Database 12c.

SQL*Plus

SQL*Plus is the primary tool for an Oracle DBA to administer the database using SQL commands. Before you can run SQL statements, you must connect to Oracle Database 12c. You can start SQL*Plus from a Windows command prompt using the SQLPLUS.EXE executable or using the \$ORACLE_HOME/bin/sqlplus executable on the Unix/Linux platform. Figure 8.2 shows connecting to SQL*Plus from a Linux workstation.

FIGURE 8.2 SQL*Plus login in Linux

```
[samuel@btlnx63 ~]$ sqlplus system

SQL*Plus: Release 12.1.0.1.0 Production on Sun Aug 25 11:49:47 2013

Copyright (c) 1982, 2013, Oracle. All rights reserved.

Enter password:
Last Successful login time: Sun Aug 25 2013 11:47:34 -05:00

Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.1.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing opt
ions

SQL> SELECT db_unique_name, cdb FROM v$database;

DB_UNIQUE_NAME          CDB
-----
C12DB1                  YES

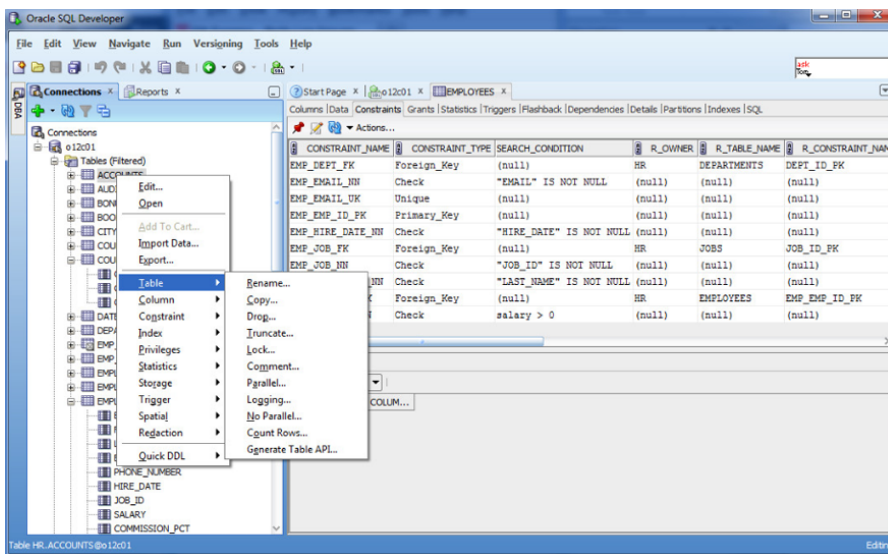
SQL> SHOW USER
USER is "SYSTEM"
SQL> █
```

To get an overview of SQL*Plus and how to connect to the database using SQL*Plus, refer to Chapter 2, “Introducing SQL.”

SQL Developer

SQL Developer is a GUI database-development tool. With SQL Developer, you can create and view the database objects, make changes to the objects, run SQL statements, run PL/SQL programs, create and edit PL/SQL programs, and perform PL/SQL debugging. SQL Developer also includes a migration utility to migrate Microsoft Access and Microsoft SQL Server databases to Oracle Database 12c. Figure 8.3 shows the object browser screen of SQL Developer.

FIGURE 8.3 The SQL Developer screen



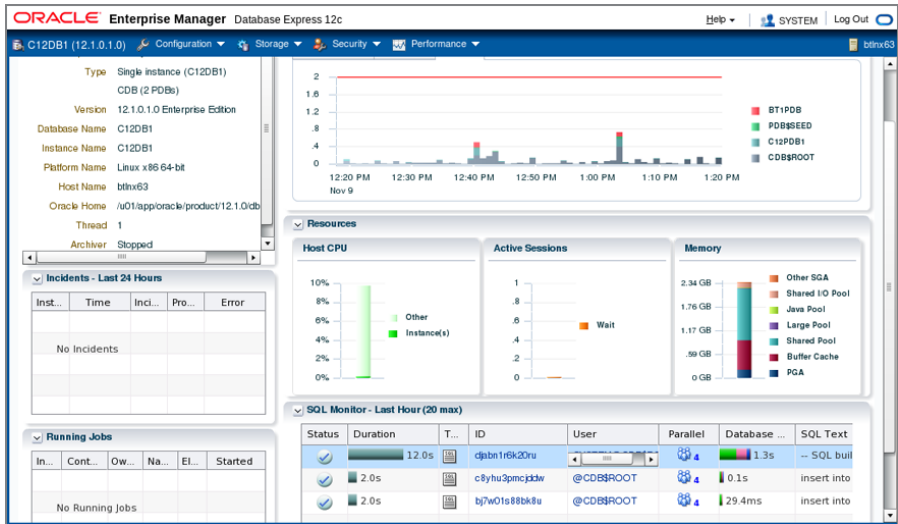
You can download and learn more about SQL Developer on the OTN website (<http://www.oracle.com/technetwork/developer-tools/sql-developer>). SQL Developer is installed by default with your Oracle Database 12c tool set.

Enterprise Manager Database Express 12c

Oracle Enterprise Manager (EM) Database Express is a web-based database management tool that is bundled with Oracle Database 12c. This is a graphical tool specifically designed to administer an Oracle database. Enterprise Manager Database Express is used to manage a single database (single instance or cluster database), whereas Enterprise Manager Cloud Control 12c can manage multiple databases and other services and applications, such as OAS,

and even non-Oracle applications at the same time. Figure 8.4 shows the Enterprise Manager Database Express home screen, where an overview of the database is shown.

FIGURE 8.4 The Enterprise Manager home screen



EM Database Express is configured using a check box in Database Configuration Assistant when you're creating a new database. EM Database Express requires that XMLDB be installed in the database. The default port configured is 5,500, and the URL for EM Database Express is `https://<hostname>:5500/em`. The port can be changed using the `DBMS_XDB_CONFIG.setHTTPsPort (<port>)` procedure.

EM Database Express is available only when the database is open. Therefore, this tool cannot be used to start or stop a database.



Oracle Enterprise Manager (OEM) Cloud Control is installed separately, outside of the Oracle Database 12c install. Agents are installed on each server that is configured in OEM Cloud Control. To learn more about OEM Cloud Control, visit <http://www.oracle.com/technetwork/oem>.

For all the database-administration examples in this chapter, you can use either SQL*Plus to perform the SQL command line or use the GUI tool Enterprise Manager Database Express. All the commands you run using SQL*Plus can also be performed using SQL Developer. However, if there are any administrative tasks that can be performed using predefined menu options in SQL Developer, we will show that. Before you start learning to administer Oracle Database 12c, let's start with the basics. In the next section, you'll learn about Oracle 12c architecture.

Oracle Database 12c Architecture

All of the previously described database administration and development tools allow users to interact with the database. Using these tools requires that user accounts be created in the database and that connectivity to the database be in place across the network. Users must also have adequate storage capacity for the data they add, and they need recovery mechanisms for restoring the transactions they are performing in the event of a hardware failure. As the DBA, you take care of each of these tasks, as well as others, which include the following:

- Selecting the server hardware on which the database software will run
- Installing and configuring the Oracle Database 12c software on the server hardware
- Deciding to use Oracle Database 12c Container or a traditional single database (now known as non-Pluggable Database (PDB)).
- Creating Oracle Database 12c database
- Creating and managing the tables and other objects used to manage the application data
- Creating and managing database users
- Establishing reliable backup and recovery procedure for the database
- Monitoring and tuning database performance
- Analyzing trends and forecasting resource and storage requirements

The remainder of this book is dedicated to helping you understand how to perform these and other important Oracle database-administration tasks. But first, to succeed as an Oracle DBA, you need to completely understand Oracle's underlying architecture and its mechanisms. Understanding the relationship between Oracle's memory structures, background processes, and I/O activities is critical before learning how to manage these areas.

The Oracle server architecture can be described in three categories:

- Server processes that communicate with users processes and interact with an Oracle instance to fulfill requests
- Logical memory structures that are collectively called an *Oracle instance*
- Physical file structures that are collectively called a *database*

You will also see how the physical structures map to the logical structures of the database you are familiar with, such as tables and indexes.

Database is a confusing term that is often used to represent different things on different platforms; the only commonality is that it is something related to storing data. In Oracle, however, the term *database* represents the physical files that store data. An *instance* is composed of the memory structures and background processes. Each database should have at least one instance associated with it. It is possible for multiple instances to access a single database; such a configuration is known as Real Application Clusters (RAC). In this book, however, you'll concentrate only on single-instance databases because RAC is not part of the OCA certification exam.

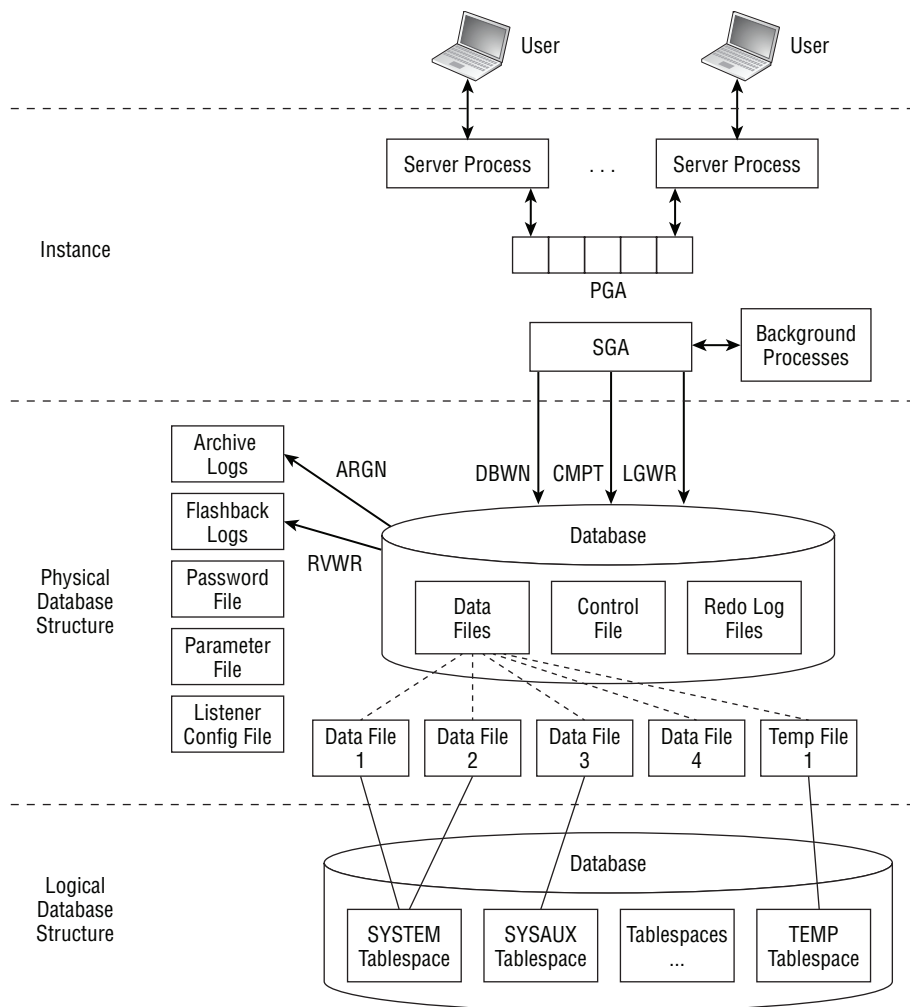
Figure 8.5 shows the parts of an Oracle instance and database at a high level. Although the architecture in Figure 8.5 may at first seem complex, each of these architecture

components is described in more detail in the following sections, beginning with the user-related processes, and is actually fairly simple. This figure is an important piece of fundamental information when learning about the Oracle Database 12c architecture.



The key database components are memory structures, process structures, and storage structures. Process and memory structures together are called an *instance*; the storage structure is called a *database*. Taken together, the instance and the database are called an *Oracle server*.

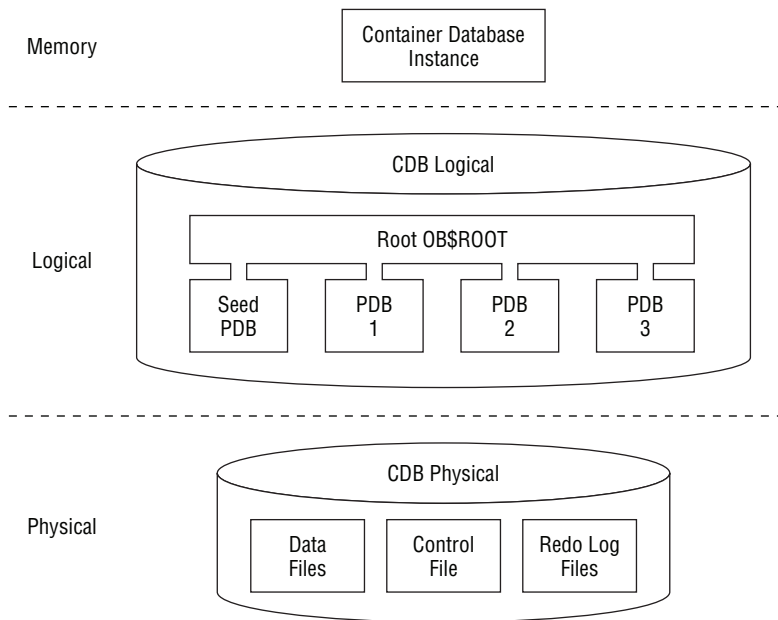
FIGURE 8.5 The Oracle Database 12c architecture



Each Oracle database consists of several schemas that reside in tablespaces. *Tablespace* is a logical storage structure at the highest level in the database. Each tablespace consists of one or more data files. The database has user data and overhead, such as database dictionary, memory, control files, archived log files, flashback files, etc. Do not worry if you do not understand these components yet; you will get to know them in the next few chapters.

Oracle Database 12c comes with a major architectural change compared to its predecessors. Oracle Database 12c allows multitenancy, meaning, you can have more than one database in a structure called a *container database*. The database overhead will be shared by all the databases in the container database. The databases in the container database are called *pluggable databases*. Administration and resource overhead are reduced by going with this architecture. Figure 8.6 shows database multitenancy.

FIGURE 8.6 Oracle Database 12c multitenancy



More on container and pluggable databases will be discussed in Chapter 9, "Creating and Operating Oracle Database 12c."

What Is a Schema?

When working with Oracle, you will often hear the words *schema* and *user* used interchangeably. Is there a difference between the two? Yes and no. A user is a defined database entity that has a set of abilities to perform activities based on their granted rights. A schema, which is associated with a user entity, is more appropriately defined as a collection of database objects. Some examples of database objects are tables, indexes, and views.

A schema can be related to a real person, such as a user of your Sales database who may have a user ID and password they use to access the database. This user may or may not own any schema objects.

Because a schema is a collection of objects, DBAs often define a schema to represent a collection of objects associated with an application. For example, a DBA might create a schema called SALES and create objects owned by that schema. Then, they can grant access to other database users who need the ability to access the SALES schema.

In this way, the schema becomes a logical collection of objects associated with an application and is not tied to any specific user. This ability makes it easy to logically group common objects that are related to specific applications or tasks using a common schema name.

The main difference is that users are the entities that perform work, and schemas are the collections of objects on which users perform work.

User and Server Processes

At the user level, two types of processes allow a user to interact with the instance and, ultimately, with the database: the *user process* and the *server process*.

Whenever a user runs an application, such as a human-resources or order-taking application, Oracle starts a user process to support the user's connection to the instance. Depending on the technical architecture of the application, the user process exists either on the user's own computer or on the middle-tier application server. The user process then initiates a connection to the instance. Oracle calls the process of initiating and maintaining communication between the user process and the instance a *connection*. Once the connection is made, the user establishes a *session* in the instance.

After establishing a session, each user starts a server process on the host server itself. It is this server process that is responsible for performing the tasks that actually allow the user to interact with the database. The server processes are allowed to interact with the instance, but not the user process directly.

Examples of these interactions include sending SQL statements to the database, retrieving needed data from the database's physical files, and returning that data to the user.



Server processes generally have a one-to-one relationship with user processes—in other words, each user process connects to one and only one server process. However, in some Oracle configurations, multiple user processes can share a single server process. We will discuss Oracle connection configurations in Chapter 12, “Understanding Oracle Network Architecture.”

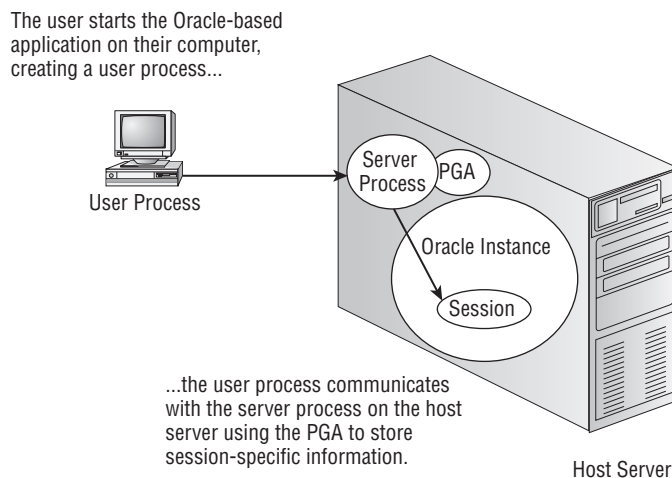
On a Unix system, it is easier to distinguish these processes. Here is an example. User initiates SQL*Plus to connect to Oracle database. You can see the process that starts SQL*Plus (user process with process ID 10604) by *samuel*. This in turn starts another process that connects to the instance (server process with process ID 10606) owned by database server user *oracle*.

```
$ ps -ef |grep sqlplus | grep -v grep
samuel  10604 10511  0 01:51 pts/2    00:00:00 sqlplus

$ ps -ef |grep 10604 | grep -v grep
samuel  10604 10511  0 01:51 pts/2    00:00:00 sqlplus
oracle  10606 10604  0 01:52 ?        00:00:00 oracleC12DB1
(DESCRIPTION=(LOCAL=YES) (ADDRESS=(PROTOCOL=beq)))
```

In addition to the user and server processes that are associated with each user connection, an additional memory structure called the *program global area* (PGA) is also created for each server process. The PGA stores user-specific session information such as bind variables and session variables. Every server process on the server has a PGA memory area. Figure 8.7 shows the relationship between a user process, server processes, and the PGA.

FIGURE 8.7 The relationship between user and server processes and the PGA



PGA memory is not shared. Each server process has a PGA associated with it and is exclusive. As a DBA, you set the total memory that can be allocated to all the PGA memory allocated to all server and background processes. The components of PGA are

SQL Work Area Area used for memory-intensive operations such as sorting or building a hash table during join operations.

Private SQL Area Holds information about SQL statement and bind variable values.

The PGA can be configured to manage automatically by setting the database parameter `PGA_AGGREGATE_TARGET`. Oracle then contains the total amount of PGA memory allocated across all database server processes and background processes within this target.

The server process communicates with the Oracle instance on behalf of the user. The Oracle instance is examined in the next section.

The Oracle Instance

An Oracle database instance consists of Oracle’s main memory structure, called the *system global area* (SGA, also known as *shared global area*) and several Oracle background processes. When the user accesses the data in the database, it is the SGA with which the server process communicates. Figure 8.8 shows the components of the SGA.

The components of the instance are described in the following sections.

Oracle Memory Structures

The SGA is a shared memory area. All the users of the database share the information maintained in this area. Oracle allocates memory for the SGA when the instance is started and de-allocates it when the instance is shut down. The SGA consists of three mandatory components and four optional components. Table 8.2 describes the required components.

TABLE 8.2 Required SGA Components

SGA Component	Description
Shared pool	Caches the most recently used SQL statements that have been issued by database users
Database buffer cache	Caches the data that has been most recently accessed by database users
Redo log buffer	Stores transaction information for recovery purposes

FIGURE 8.8 SGA components

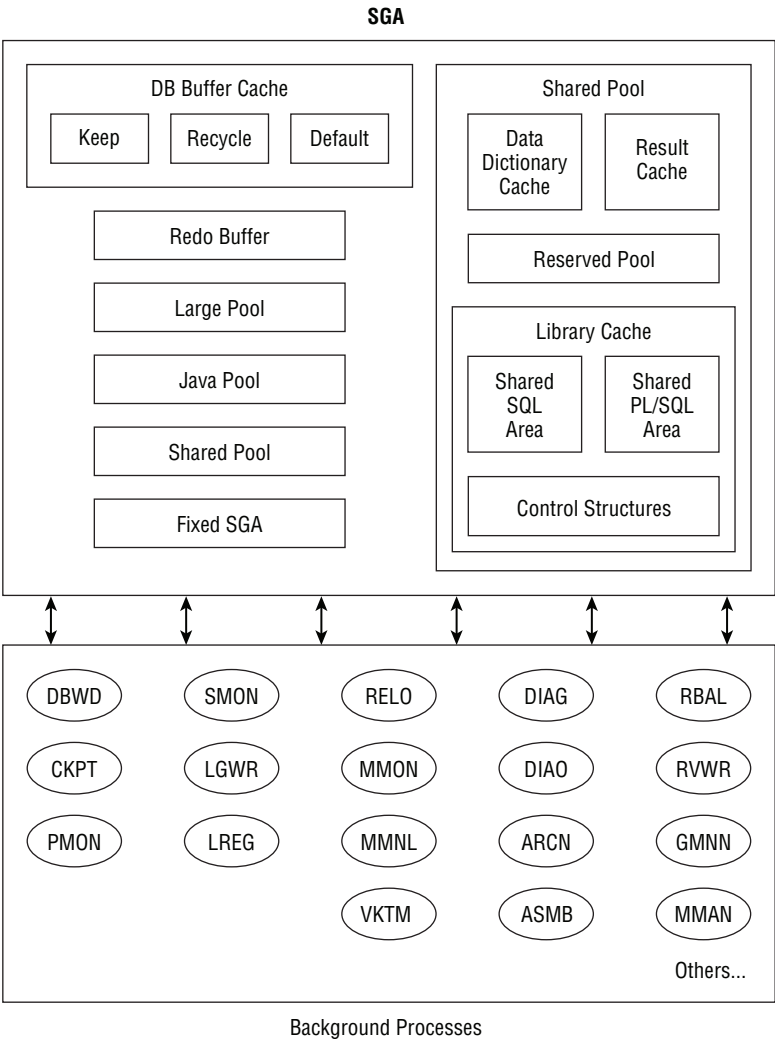


Table 8.3 describes the optional SGA components required, based on the database configuration and use.

TABLE 8.3 Optional SGA Components

SGA Component	Description
Java pool	Caches the most recently used Java objects and application code when Oracle’s JVM option is used.
Large pool	Caches data for large operations such as Recovery Manager (RMAN) backup and restore activities and Shared Server components.
Streams pool	Caches the data associated with queued message requests when Oracle’s Advanced Queuing option is used.
Result cache	This new area is introduced in Oracle Database 12c and stores results of SQL queries and PL/SQL functions for better performance.

Oracle Database 12c can manage the components of SGA and PGA automatically using the Automatic Memory Management (AMM) feature. Memory in the SGA is allocated in units of contiguous memory called *granules*. The size of a granule depends on the parameter `MEMORY_MAX_TARGET`. If `MEMORY_MAX_TARGET` is larger than 1,024MB, the granule size is either 16MB or 4MB. `MEMORY_MAX_TARGET` is discussed in detail in Chapter 14, “Maintaining the Database and Managing Performance.” A minimum of three granules must be allocated to SGA—one each for the required components in Table 8.2.

The sizes of these SGA components can be managed in two ways: manually or automatically. If you choose to manage these components manually, you must specify the size of each SGA component and then increase or decrease the size of each component according to the needs of the application. If these components are managed automatically, the instance itself will monitor the utilization of each SGA component and adjust their sizes accordingly, relative to a predefined maximum allowable aggregate SGA size.

Oracle Database 12c provides several dynamic performance views to see the components and sizes of SGA; you can use `V$SGA` and `V$SGAINFO`, as shown here:

```
SQL> select * from v$sga;
```

NAME	VALUE	CON_ID
-----	-----	-----
Fixed Size	2290368	0
Variable Size	620760384	0
Database Buffers	1694498816	0
Redo Buffers	20762624	0

Alternatively, you may use the `SHOW SGA` command from SQL*Plus, as shown here:

```
SQL> show sga
```

```

Total System Global Area 2338312192 bytes
Fixed Size                2290368 bytes
Variable Size             620760384 bytes
Database Buffers         1694498816 bytes
Redo Buffers              20762624 bytes
SQL>

```

The output from this query shows that the total size of the SGA is 2,338,312,192 bytes. This total size is composed of the variable space that is composed of the shared pool, the large pool, and the Java pool (620,760,384 bytes); the database buffer cache (1,694,498,816 bytes); the redo log buffers (20,762,624 bytes); and some additional space (2,290,368 bytes) that stores information used by the instance's background processes. The V\$SGAINFO view displays additional details about the allocation of space within the SGA, as shown in the following query:

```
SQL> SELECT * FROM v$sgainfo;
```

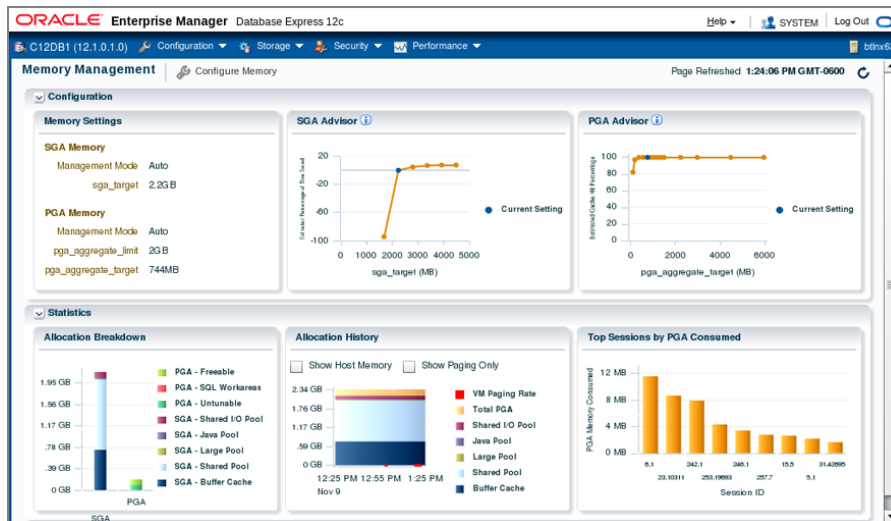
NAME	BYTES	RES	CON_ID
-----	-----	---	-----
Fixed SGA Size	2290368	No	0
Redo Buffers	20762624	No	0
Buffer Cache Size	1694498816	Yes	0
Shared Pool Size	570425344	Yes	0
Large Pool Size	33554432	Yes	0
Java Pool Size	16777216	Yes	0
Streams Pool Size	0	Yes	0
Shared IO Pool Size	117440512	Yes	0
Data Transfer Cache Size	0	Yes	0
Granule Size	16777216	No	0
Maximum SGA Size	2338312192	No	0
Startup Overhead in Shared Pool	149697048	No	0
Free SGA Memory Available	0		0

```
13 rows selected.
```

```
SQL>
```

The results of this query show in detail how much space is occupied by each component in the shared pool. The components with the RESIZEABLE column with a value of Yes can be managed dynamically by Oracle Database 12c.

You can also use EM Database Express to view the sizes of each of the SGA components, as shown in Figure 8.9. From the home screen, go to the Server tab and click Memory Advisors to see this.

FIGURE 8.9 EM Database Express showing SGA components

You'll learn more about the components in the SGA in the next sections.

Database Buffer Cache

The *database buffer cache* is the area in SGA that caches the database data, holding blocks from the data files that have been accessed recently. The database buffer cache is shared among all the users connected to the database. There are three types of buffers:

- *Dirty buffers* are the buffer blocks that need to be written to the data files. The data in these buffers has changed and has not yet been written to the disk.
- *Free buffers* do not contain any data or are free to be overwritten. When Oracle reads data from the disk, free buffers hold this data.
- *Pinned buffers* are the buffers that are currently being accessed or explicitly retained for future use.

Oracle uses a *least recently used algorithm* (LRU algorithm) to manage the contents of the shared pool and database buffer cache. When a user's server process needs to put a SQL statement into the shared pool or copy a database block into the buffer cache, Oracle uses the space in memory that is occupied by the least recently accessed SQL statement or buffer to hold the requested SQL or block copy. Using this technique, Oracle keeps frequently accessed SQL statements and database buffers in memory longer, improving the overall performance of the server by minimizing parsing and physical disk I/O.



The background process **DBWn** writes the database blocks from the database buffer cache to the data files. Dirty buffers contain data that changed and must be written to disk.

To better manage the buffer cache better, Oracle Database 12c provides three buffer caches. The DEFAULT cache is the default and is required. The KEEP cache and the RECYCLE cache can be optionally configured. By default all the data read from the disk is written to the DEFAULT pool. If you want certain data not to be aged from memory, you can configure the KEEP pool and use the ALTER TABLE statement to specify which tables should use the KEEP pool. Similarly, if you do not want to age out good data from the default cache for temporary data, you may specify such tables to have the RECYCLE pool instead of the default. The blocks in the KEEP pool also follow the LRU algorithm to age out blocks when new blocks need space in the KEEP pool. By sizing the KEEP pool appropriately, you can hold frequently used blocks longer in the KEEP pool. The RECYCLE cache removes the buffers from memory as soon as they are no longer needed.



The DB_CACHE_SIZE parameter specifies the size of the database buffer cache DEFAULT pool. To configure the KEEP and RECYCLE pools, use the DB_KEEP_CACHE_SIZE and DB_RECYCLE_CACHE_SIZE parameters.

Since Oracle Database 11g Release 2, Oracle Linux and Oracle Solaris servers make use of the flash storage for additional buffer cache. *Database Smart Flash Cache* allows the database buffer cache to be expanded beyond the SGA in main memory to a second level cache on flash memory. When the block expires from the SGA buffer cache, it is evicted to the database flash cache until required again. Flash cache is configured using two database parameters. DB_FLASH_CACHE_FILE identifies the flash device, and DB_FLASH_CACHE_SIZE specifies the size of flash cache.

Redo Log Buffer

The *redo log buffer* is a circular buffer in the SGA that holds information about the changes made to the database data. The changes are known as *redo entries* or *change vectors* and are used to redo the changes in case of a failure. DML and DDL statements are used to make changes to the database data. The parameter LOG_BUFFER determines the size of the redo log buffer cache.



The background process LGWR writes the redo log information to the online redo log files.

Shared Pool

The *shared pool* portion of the SGA holds information such as SQL, PL/SQL procedures and packages, the data dictionary, locks, character-set information, security attributes, and so on. The shared pool consists of the library cache and the data dictionary cache.

The *library cache* contains the shared SQL areas, private SQL areas, PL/SQL programs, and control structures such as locks and library cache handles.

The *shared SQL area* is used for maintaining recently executed SQL statements and their execution plans. Oracle divides each SQL statement that it executes into a shared SQL area

and a private SQL area. When two users are executing the same SQL, the information in the shared SQL area is used for both. The shared SQL area contains the parse tree and execution plan, whereas the private SQL area contains values for the bind variables (persistent area) and runtime buffers (runtime area). Oracle creates the runtime area as the first step of an execute request. For INSERT, UPDATE, and DELETE statements, Oracle frees the runtime area after the statement has been executed. For queries, Oracle frees the runtime area only after all rows have been fetched or the query has been canceled.

Oracle processes PL/SQL program units the same way it processes SQL statements. When a PL/SQL program unit is executed, the code is moved to the *shared PL/SQL area*, and the individual SQL commands within the program unit are moved to the shared SQL area. Again, the shared program units are maintained in memory with an LRU algorithm.

The third area in the library cache is used to store control information and is maintained internally by Oracle. Various locks, latches, and other control structures reside here, and any server process that requires this information can access it.

The *data dictionary cache* holds the most recently used database dictionary information. The data dictionary cache is also known as the *row cache* because it holds data as rows instead of buffers (which hold entire blocks of data).

The SQL query *result cache* stores the results of queries. If an application runs the same SELECT statement repeatedly and if the results are cached, then the database can return them immediately. In this way, the database avoids the expensive operation of rereading blocks to show results.

The PL/SQL function *result cache* is used to hold the SQL and PL/SQL function results. Executions of similar SQL statements can use the cached results to answer query requests. Because retrieving results from the SQL query result cache is faster than rerunning a query, frequently run queries experience a significant performance improvement when their results are cached.

The *reserved pool* is an area in the shared pool used to allocate large chunks of memory. Its size is determined by the SHARED_POOL_RESERVED_SIZE initialization parameter.



The parameter SHARED_POOL_SIZE determines the size of the shared pool.

Large Pool

The *large pool* is an optional area in the SGA that the DBA can configure to provide large memory allocations for specific database operations such as an RMAN backup or restore. The large pool allows Oracle to request large memory allocations from a separate pool to prevent contention from other applications for the same memory. The large pool does not have an LRU list; Oracle Database 12c does not attempt to age objects out of the large pool. The parameter LARGE_POOL_SIZE determines the size of the large pool.

Java Pool

The *Java pool* is another optional area in the SGA that the DBA can configure to provide memory for Java operations, just as the shared pool is provided for processing SQL and PL/SQL statements. The parameter `JAVA_POOL_SIZE` determines the size of the Java pool.

Streams Pool

The *streams pool* is exclusively used by Oracle streams. The `STREAMS_POOL_SIZE` parameter determines the size of the streams pool.



If any SGA component size is set smaller than the granule size, the size of the component is rounded to the nearest granule size.



Oracle Database 12c can manage all the components of the SGA and PGA automatically; there is no need for the DBA to configure each pool individually. You will learn more about automatic memory management in Chapter 14.

Oracle Background Process

Many types of Oracle background processes exist, designed specifically for different functions. Each process performs a specific job in helping to manage an instance. Five Oracle background processes are required by the Oracle instance, and several background processes are optional. The required background processes are found in all Oracle instances. Optional background processes may or may not be used, depending on the features that are being used in the database. Table 8.4 describes the required background processes. The database instance terminates abruptly if you terminate any of these processes (except RECO, DIA0, DIAG—these are restarted automatically if the process dies or is terminated) or if there is an error in one of these processes and Oracle had to shut down the process. The processes are started by default when the instance starts.

TABLE 8.4 Required Oracle Background Processes

Process Name	OS Process	Description
Database Writer	DBWn BWnn	Writes modified database blocks from the SGA's database buffer cache to the data files on disk.
Checkpoint	CKPT	Updates the data file headers following a checkpoint event.

TABLE 8.4 Required Oracle Background Processes *(continued)*

Process Name	OS Process	Description
Log Writer	LGWR	Writes transaction recovery information from the SGA's redo log buffer to the online redo log files on disk.
Process Monitor	PMON	Cleans up failed user database connections.
System Monitor	SMON	Performs instance recovery following an instance crash, coalesces free space in the database, and manages space used for sorting.
Listener Registration	LREG	Registers information about the database instance and dispatcher processes with the listener.
Recoverer	RECO	Recovers failed transactions that are distributed across multiple databases when using Oracle's distributed database feature.
Memory Monitor	MMON	Gathers and analyzes statistics used by the Automatic Workload Repository feature. See Chapter 14 for more information on using this feature.
Memory Monitor Light	MMNL	Gathers and analyzes statistics used by the Active Session History feature. See Chapter 14 for more information on using this feature.
Virtual Keeper of Time	VKTM	Responsible for providing a wall-clock time (updated every second) and reference-time counter.
Diagnosability	DIAG	Performs diagnostic dumps.
Diagnosability	DIA0	Diagnostic process responsible for hang detection and deadlock resolution.

Table 8.5 describes some of the optional background processes.

TABLE 8.5 Optional Oracle Background Processes

Process Name	OS Process	Description
Archiver	ARC <i>n</i>	Copies the transaction recovery information from the redo log files to the archive location. Nearly all production databases use this optional process. You can have up to 30 archival processes (ARC0–ARC9, ARCa–ARCt).
Recovery Writer	RVWR	Writes flashback data to flashback database logs in the fast recovery area.
ASM Disk	ASMB	Present on databases using Automatic Storage Management disks.
ASM Balance	RBAL	Coordinates rebalance activity of disks in an ASM disk group.
Job Queue Monitor	CJQ <i>n</i>	Assigns jobs to the job queue processes when using Oracle's job scheduling feature.
Job Queue	J <i>nnn</i>	Executes database jobs that have been scheduled using Oracle's job-scheduling feature.
Queue Monitor	QM <i>Nn</i>	Monitors the messages in the message queue when Oracle's Advanced Queuing feature is used.
Event Monitor	EMNC	Process responsible for event-management coordination and notification.
Flashback Data Archive	FBDA	Archives historical records from a table when the flashback data archive feature is used.
Parallel Query Slave	Q <i>nnn</i>	Carries out portions of a larger overall query when Oracle's Parallel Query feature is used.
Dispatcher	D <i>nnn</i>	Assigns user's database requests to a queue where they are then serviced by shared server processes when Oracle's Shared Server feature is used. See Chapter 11, "Managing Data and Undo," for details on using shared servers.
Shared Server	S <i>nnn</i>	Server processes that are shared among several users when Oracle's Shared Server feature is used. See Chapter 11 for details on using shared servers.

TABLE 8.5 Optional Oracle Background Processes *(continued)*

Process Name	OS Process	Description
Memory Manager	MMAN	Manages the size of each individual SGA component when Oracle's Automatic Shared Memory Management feature is used. See Chapter 14 for more information on using this feature.
Recovery Writer	RVWR	Writes recovery information to disk when Oracle's Flashback Database feature is used. See Chapter 16, "Implementing Database Backups," for details on how to use the Flashback Database feature.
Change Tracking Writer	CTWR	Keeps track of which database blocks have changed when Oracle's incremental Recovery Manager feature is used. See Chapter 16 for details on using Recovery Manager to perform backups.
Space Management Coordinator	SMCO	Coordinates various space management tasks. Worker processes are identified with Wnnn.

On Unix systems, you can view these background processes from the operating system using the `ps` command, as shown here:

```
$ ps -ef |grep C12DB1
oracle 3623 1 0 Aug24 ? 00:00:05 ora_pmon_C12DB1
oracle 3625 1 0 Aug24 ? 00:00:06 ora_psp0_C12DB1
oracle 3627 1 2 Aug24 ? 00:12:21 ora_vktm_C12DB1
oracle 3631 1 0 Aug24 ? 00:00:01 ora_gen0_C12DB1
oracle 3633 1 0 Aug24 ? 00:00:01 ora_mman_C12DB1
oracle 3637 1 0 Aug24 ? 00:00:01 ora_diag_C12DB1
oracle 3639 1 0 Aug24 ? 00:00:01 ora_dbrm_C12DB1
oracle 3641 1 0 Aug24 ? 00:00:29 ora_dia0_C12DB1
oracle 3643 1 0 Aug24 ? 00:00:12 ora_dbw0_C12DB1
oracle 3645 1 0 Aug24 ? 00:00:06 ora_lgwr_C12DB1
oracle 3647 1 0 Aug24 ? 00:00:10 ora_ckpt_C12DB1
oracle 3649 1 0 Aug24 ? 00:00:12 ora_lg00_C12DB1
oracle 3651 1 0 Aug24 ? 00:00:04 ora_lg01_C12DB1
oracle 3653 1 0 Aug24 ? 00:00:01 ora_smon_C12DB1
oracle 3655 1 0 Aug24 ? 00:00:00 ora_reco_C12DB1
oracle 3657 1 0 Aug24 ? 00:00:01 ora_lreg_C12DB1
oracle 3659 1 0 Aug24 ? 00:00:23 ora_mmon_C12DB1
oracle 3661 1 0 Aug24 ? 00:00:20 ora_mmn1_C12DB1
```

oracle	3663	1	0	Aug24	?	00:00:00	ora_d000_C12DB1
oracle	3679	1	0	Aug24	?	00:01:39	ora_p000_C12DB1
oracle	3681	1	0	Aug24	?	00:01:16	ora_p001_C12DB1
oracle	3683	1	0	Aug24	?	00:00:00	ora_tmon_C12DB1
oracle	3685	1	0	Aug24	?	00:00:01	ora_tt00_C12DB1
oracle	3687	1	0	Aug24	?	00:00:00	ora_smco_C12DB1
oracle	3691	1	0	Aug24	?	00:00:00	ora_aqpc_C12DB1
oracle	3693	1	0	Aug24	?	00:00:03	ora_w000_C12DB1
oracle	3697	1	0	Aug24	?	00:00:15	ora_p002_C12DB1
oracle	3699	1	0	Aug24	?	00:00:14	ora_p003_C12DB1
oracle	3701	1	0	Aug24	?	00:00:01	ora_p004_C12DB1
oracle	3703	1	0	Aug24	?	00:00:01	ora_p005_C12DB1
oracle	3705	1	0	Aug24	?	00:00:00	ora_p006_C12DB1
oracle	3707	1	0	Aug24	?	00:00:00	ora_p007_C12DB1
oracle	3737	1	0	Aug24	?	00:00:00	ora_qm02_C12DB1
oracle	3741	1	0	Aug24	?	00:00:00	ora_q002_C12DB1
oracle	3743	1	0	Aug24	?	00:00:00	ora_q003_C12DB1
oracle	3746	1	0	Aug24	?	00:00:27	ora_cjq0_C12DB1
oracle	3951	1	0	Aug24	?	00:00:03	ora_w001_C12DB1
oracle	4102	1	0	Aug24	?	00:00:04	ora_w002_C12DB1
oracle	4886	1	0	Aug24	?	00:00:00	ora_w003_C12DB1
oracle	4906	1	0	Aug24	?	00:00:00	ora_w004_C12DB1
oracle	6504	1	0	Aug24	?	00:00:05	ora_s000_C12DB1
oracle	11619	1	0	03:13	?	00:00:00	ora_p00a_C12DB1
oracle	11621	1	0	03:13	?	00:00:00	ora_p00b_C12DB1
oracle	11632	1	0	03:14	?	00:00:00	ora_p008_C12DB1
oracle	11634	1	0	03:14	?	00:00:00	ora_p009_C12DB1

This output shows that several background processes are running on the Linux server for the C12DB1 database, which is a container database.

Threaded Execution

A new parameter introduced in Oracle Database 12c allows multiple background processes to share a single OS process on Unix, similar to the model Oracle has on Windows. This behavior on Unix is controlled by the parameter `THREADED_EXECUTION`, which by default is set to `FALSE`. The multithreaded Oracle Database model enables Oracle processes to execute as operating system threads in separate address spaces. In default process models, `SPID` and `STID` columns of `V$PROCESS` will have the same values, whereas in multithreaded models, each `SPID` (process) will have multiple `STID` (threads) values. The `EXECUTION_TYPE` column in `V$PROCESS` will show *THREAD*.

The dynamic view `V$BGPROCESS` shows the background processes available. The following query may be used to list all the background processes running on the instance.

```
SQL> SELECT min(name || ': ' || description) process_description
2 FROM v$bgprocess
3 group by substr(name,1,3)
4* ORDER BY 1
SQL> /
```



On Windows systems, each background and server process is a thread to the `oracle.exe` process.

Knowing the purpose of the required background processes is a must for the OCA certification exam. We'll discuss those purposes in the next subsections.

Database Writer (DBW_n)

The purpose of the *database writer process* (DBW_n) is to write the contents of the dirty buffers to the data files. By default, Oracle starts one database writer process when the instance starts. For multiuser and busy systems, you can have up to 100 database writer processes to improve performance. The names of the first 36 database writer processes are DBW0–DBW9 and DBWa–DBWz. The names of the 37th through 100th database writer processes are BW36–BW99. The parameter `DB_WRITER_PROCESSES` determines the additional number of database writer processes to be started. Having more DBW_n processes than the number of CPUs is normally not beneficial.

The DBW_n process writes the modified buffer blocks to disk, so more free buffers are available in the buffer cache. Writes are always performed in bulk to reduce disk contention; the number of blocks written in each I/O is OS-dependent.

Checkpoint (CKPT)

When a change is committed to a database, Oracle identifies the transaction with a unique number called the system change number (SCN). The value of an SCN is the logical point in time at which changes are made to a database. A *checkpoint* is when the DBW_n process writes all the dirty buffers to the data files. When a checkpoint occurs, Oracle must update the control file and each data file header to record the checkpoint. This update is done by the *checkpoint process* (CKPT); the DBW_n process writes the actual data blocks to the data files.

Checkpoints help reduce the time required for instance recovery. If checkpoints occur too frequently, disk contention becomes a problem with the data file updates. If checkpoints occur too infrequently, the time required to recover a failed database instance can be significantly longer. Checkpoints occur automatically when an online redo log file is full (a log switch happens).

When a redo log switch happens, the checkpoint process needs to update the header of all the data files; this causes performance issues on databases with hundreds of data files. To alleviate this situation, Oracle uses incremental checkpoints. Here the responsibility of

updating the data file header is given to the DBW n process, when it writes dirty buffers to data files. The CKPT process updates only the control file with the checkpoint position, not the data files.

When Does Database Writer Write?

The DBW n background process writes to the data files whenever one of the following events occurs:

- A user's server process has searched too long for a free buffer when reading a buffer into the buffer cache.
- The number of modified and committed, but unwritten, buffers in the database buffer cache is too large.
- At a database checkpoint event. See Chapter 16 for information on checkpoints.
- The instance is shut down using any method other than a shutdown abort.
- A tablespace is placed into backup mode.
- A tablespace is taken offline to make it unavailable or is changed to READ ONLY.
- A segment is dropped.

A *database checkpoint* or *thread checkpoint* is when all data file headers as well as the control file are updated with checkpoint information. At this time, the database writes all the dirty buffers to data files. This happens during normal database shutdown, online redo log switch, forced checkpoint using ALTER SYSTEM CHECKPOINT, or when the database is placed in backup mode using ALTER DATABASE BEGIN BACKUP.

Log Writer (LGWR)

The *log writer process* (LGWR) writes the blocks in the redo log buffer of the SGA to the online redo log files. When the LGWR writes log buffers to disk, Oracle server processes can write new entries in the redo log buffer. LGWR writes the entries to the disk fast enough to ensure that room is available for the server process to write the redo entries. There can be only one LGWR process in the database.

If the redo log files are multiplexed, LGWR writes simultaneously to all the members of the redo log group. Even if one of the log files in the group is damaged, LGWR writes the redo information to the available files. LGWR writes to the redo log files sequentially so that transactions can be applied in order in the event of a failure.

As soon as a transaction commits, the information is written to redo log files. By writing the committed transaction immediately to the redo log files, the change to the database is never lost. Even if the database crashes, committed changes can be recovered from the online redo log files and applied to the data files.

When Does Log Writer Write?

The LGWR background process writes to the current redo log group under any of the following conditions:

- Three seconds since the last LGWR write
- When a user commits a transaction
- When the redo log buffer is a third full
- When the redo log buffer contains 1MB worth of redo information
- Whenever a database checkpoint occurs

Process Monitor (PMON)

The *process monitor process* (PMON) cleans up failed user processes and frees up all the resources used by the failed process. It resets the status of the active transaction table and removes the process ID from the list of active processes. It reclaims all the resources held by the user and releases all locks on tables and rows held by the user. PMON wakes up periodically to check whether it is needed. Other processes can call PMON if they detect a need for a PMON process.

PMON also checks on some optional background processes and restarts them if any have stopped.

System Monitor (SMON)

The *system monitor process* (SMON) performs instance or crash recovery at database startup by using the online redo log files. SMON is also responsible for cleaning up temporary segments in the tablespaces that are no longer used and for coalescing the contiguous free space in the dictionary-managed tablespaces. If any dead transactions were skipped during instance recovery because of file-read or offline errors, SMON recovers them when the tablespace or data file is brought back online. SMON wakes up regularly to check whether it is needed. Other processes can call SMON if they detect a need for an SMON process.



In Windows environments, a Windows service called `OracleServiceInstanceName` is also associated with each instance. This service must be started in order to start up the instance in Windows environments.

Oracle Storage Structures

An instance is a memory structure, but the Oracle database consists of a set of physical files that reside on the host server's disk drives. The physical storage structures include three

types of files. These files are called *control files*, *data files*, and *redo log files*. The additional physical files that are associated with an Oracle database but are not technically part of the database are as follows: the *password file*, the *parameter file*, and any *archived redo log files*. The Oracle Net configuration files are also required for connectivity to an Oracle database. To roll back database changes using the Database Flashback feature, the *flashback log files* are used. Table 8.6 summarizes the role that each of these files plays in the database architecture.

TABLE 8.6 Oracle Physical Files

File Type	Information Contained in Files
Control file	Locations of other physical files, database name, database block size, database character set, and recovery information. These files are required to open the database.
Data file	All application data and internal metadata.
Redo log file	Record of all changes made to the database; used for instance recovery.
Parameter (pfile or spfile)	Configuration parameters for the SGA, optional Oracle features, and background processes.
Archived redo log file	Copy of the contents of online redo logs, used for database recovery and for change capture.
Password file	Optional file used to store names of users who have been granted the SYSDBA and SYSOPER privileges. See Chapter 13, “Implementing Security and Auditing,” for details on SYSDBA and SYSOPER privileges.
Oracle Net file	Entries that configure the database listener and client-to-database connectivity. See Chapter 12 for details.
Flashback log file	If the database has flashback logging enabled, files are written to the fast recovery area.

Figure 8.10 shows where to view the physical storage information of the database using OEM Database Express 12c.

Figure 8.11 shows how to access the DBA menu in SQL Developer. Under DBA menu, you can view and administer several components of the database.

The three types of critical files that make up a database—the control file, the data file, and the redo log file—are described in the following sections.

FIGURE 8.10 The OEM Database Express Storage menu

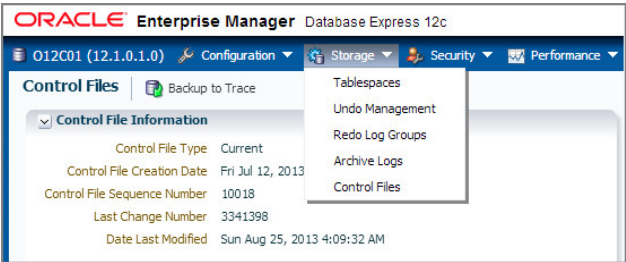
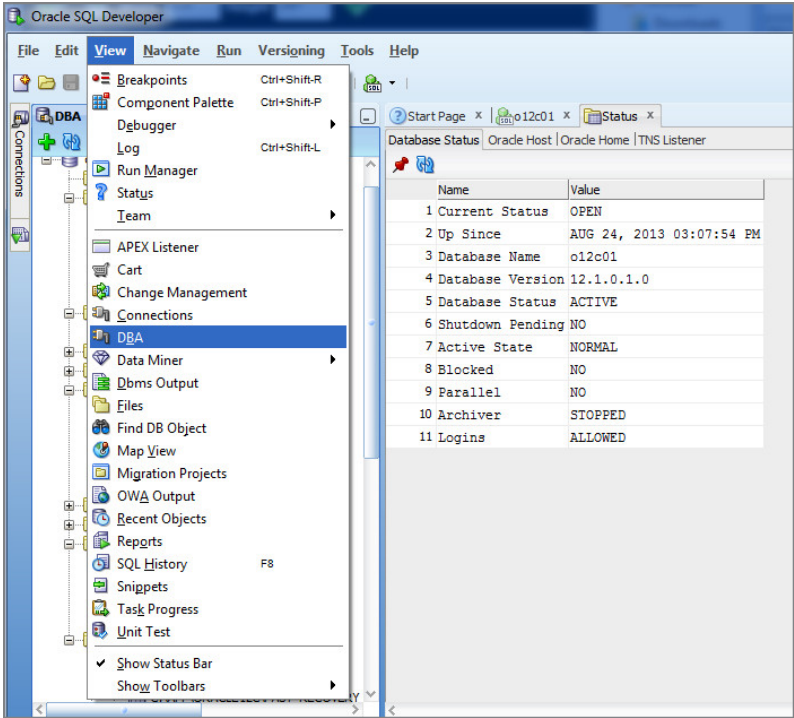


FIGURE 8.11 The SQL Developer DBA menu



Control Files

Control files are critical components of the database because they store important information that is not available anywhere else. This information includes the following:

- The name of the database
- A database-creation timestamp

- The names, locations, and sizes of the data files and redo log files
- Tablespace information
- Redo log information used to recover the database in the case of a disk failure or user error
- Archived log information
- RMAN backup information
- Checkpoint information

The following query shows the types of information kept in the control file, indicating the importance of this file.

```
SQL> SELECT type FROM v$controlfile_record_section;
```

```
TYPE
```

```
-----
DATABASE
CKPT PROGRESS
REDO THREAD
REDO LOG
DATAFILE
FILENAME
TABLESPACE
TEMPORARY FILENAME
RMAN CONFIGURATION
LOG HISTORY
OFFLINE RANGE
ARCHIVED LOG
BACKUP SET
BACKUP PIECE
BACKUP DATAFILE
BACKUP REDOLOG
DATAFILE COPY
BACKUP CORRUPTION
COPY CORRUPTION
DELETED OBJECT
PROXY COPY
BACKUP SPFILE
DATABASE INCARNATION
FLASHBACK LOG
RECOVERY DESTINATION
INSTANCE SPACE RESERVATION
REMOVABLE RECOVERY FILES
RMAN STATUS
THREAD INSTANCE NAME MAPPING
MTTR
DATAFILE HISTORY
STANDBY DATABASE MATRIX
GUARANTEED RESTORE POINT
RESTORE POINT
DATABASE BLOCK CORRUPTION
ACM OPERATION
FOREIGN ARCHIVED LOG
PDB RECORD
AUXILIARY DATAFILE COPY
MULTI INSTANCE REDO APPLY
PDBINC RECORD
```

```
41 rows selected.
```

```
SQL>
```

The control files are created when the database is created in the locations specified in the `control_files` parameter in the parameter file. Because a loss of the control files negatively impacts the ability to recover the database, most databases multiplex their control files to multiple locations. Oracle uses the CKPT background process to automatically update each of these files as needed, keeping the contents of all copies of the control synchronized. You can use the dynamic performance view `V$CONTROLFILE` to display the names and locations of all the database’s control files. A sample query on `V$CONTROLFILE` is shown here:


```
SQL> SELECT name FROM v$controlfile;
```

NAME

/u01/app/oracle/oradata/C12DB1/controlfile/o1_mf_8tx7cfnl_.ctl
/u01/app/oracle/fast_recovery_area/C12DB1/controlfile/o1_mf_8tx7cfz0_.ctl

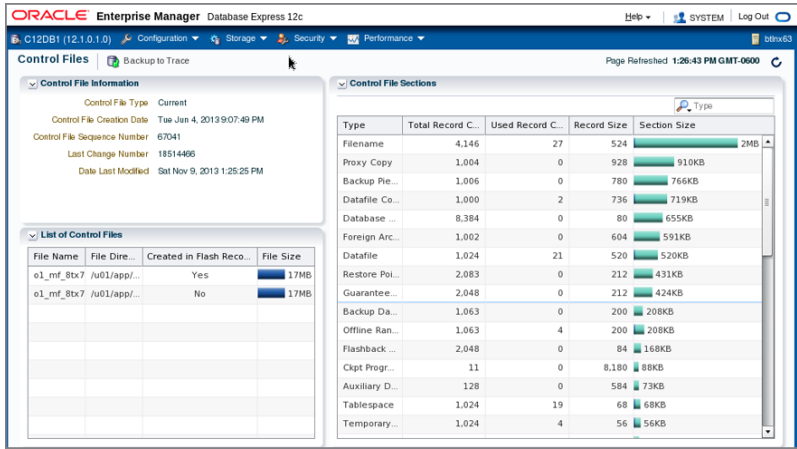
SQL>

This query shows that the database has two control files, called `o1_mf_8tx7cfnl_.ctl` and `o1_mf_8tx7cfz0_.ctl`, which are stored in different directories. The control files can be stored in any directory; however, it is better if they are physically stored on different disks. You can also monitor control files using EM Database Express (on the Server tab, choose Control Files under Storage, as shown in Figure 8.12).



Control files are usually the smallest files in the database, generally a few megabytes in size. However, they can be larger, depending on the `PFIL`/`SPFILE` setting for `CONTROLFILE_RECORD_KEEP_TIME` when the Recovery Manager feature is used.

FIGURE 8.12 EM Database Express showing control files



In the database, the control files keep track of the names, locations, and sizes of the database data files. Data files, and their relationship to another database structure called a *tablespace*, are examined in the next section.

Data Files

Data files are the physical files that actually store the data that has been inserted into each table in the database. The size of the data files is directly related to the amount of table data they store. Data files are the physical structure behind another database storage area called a *tablespace*. A tablespace is a logical storage area within the database. Tablespaces group logically related segments. For example, all the tables for the Accounts Receivable application might be stored together in a tablespace called AR_TAB, and the indexes on these tables might be stored in a tablespace called AR_IDX.

By default, every Oracle Database 12c must have at least three tablespaces. Table 8.7 describes these tablespaces.

TABLE 8.7 Required Tablespaces in Oracle 12c

Tablespace Name	Description
SYSTEM	Stores the data dictionary tables and PL/SQL code.
SYSAUX	Stores segments used for database options such as the Automatic Workload Repository, Online Analytical Processing (OLAP), and Spatial.
TEMP	Used for performing large sort operations. TEMP is required when the SYSTEM tablespace is created as a locally managed tablespace; otherwise, it is optional. See Chapter 10, “Understanding Storage and Space Management,” for details.

In addition to these three required tablespaces, most databases have tablespaces for storing other database segments such as undo and application data. Many production databases often have many more tablespaces for storing application segments. Either you or the application vendor determines the total number and names of these tablespaces. Tablespaces are discussed in detail in Chapter 10, “Understanding Storage and Space Management.”

For each tablespace in the database, there must be at least one data file. Some tablespaces may be composed of several data files for management or performance reasons. The data dictionary view DBA_DATA_FILES shows the data files associated with each tablespace in the database. The following SQL statement shows a sample query on the DBA_DATA_FILES data dictionary view:

```
SQL> SELECT tablespace_name, file_name
2    FROM dba_data_files
3   ORDER BY tablespace_name;
```

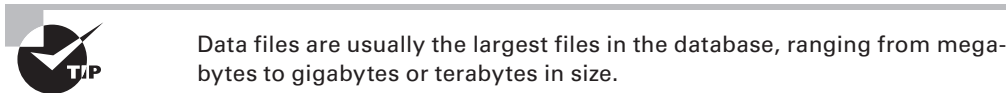
TABSPACE_N FILE_NAME

```
-----  
APPL_DATA      /u01/app/oracle/oradata/12cR1/appl_data01.dbf  
APPL_DATA      /u01/app/oracle/oradata/12cR1/appl_data02.dbf  
EXAMPLE        /u01/app/oracle/oradata/12cR1/example01.dbf  
SYSAUX         /u01/app/oracle/oradata/12cR1/sysaux01.dbf  
SYSTEM         /u01/app/oracle/oradata/12cR1/system01.dbf  
UNDOTBS1       /u01/app/oracle/oradata/12cR1/undotbs01.dbf  
USERS          /u01/app/oracle/oradata/12cR1/users01.dbf
```

7 rows selected.

SQL>

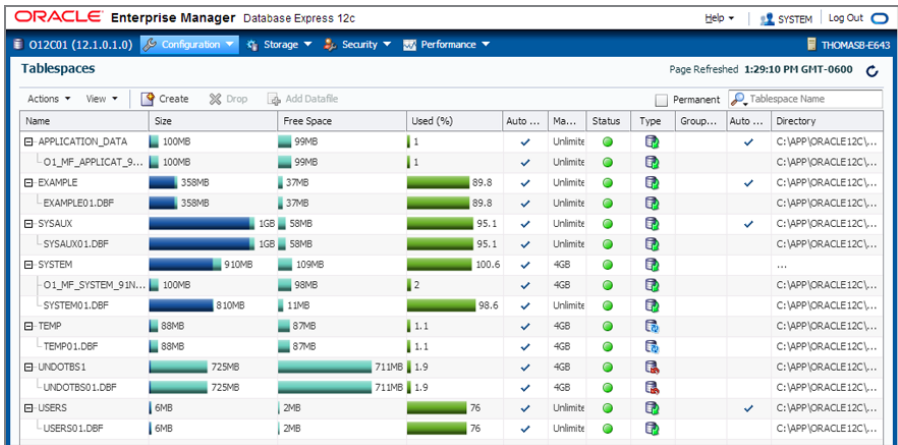
The output shows that the APPL_DATA tablespace is comprised of two data files; all other tablespaces have one data file. You can also monitor data files using EM, as shown in Figure 8.13.



Data files are usually the largest files in the database, ranging from megabytes to gigabytes or terabytes in size.

When a user performs a SQL operation on a table, the user's server process copies the affected data from the data files into the database buffer cache in the SGA. If the user has performed a committed transaction that modifies that data, the database writer process (DBWn) ultimately writes the modified data back to the data files.

FIGURE 8.13 EM Database Express showing data files



Redo Log Files

Whenever a user performs a transaction in the database, the information needed to reproduce this transaction in the event of a database failure is written to the redo log files, and the user does not get a confirmation of the commit until the transaction is successfully written to the redo log files.

Because of the important role that redo logs play in Oracle's recovery mechanism, they are usually multiplexed. This means that each redo log contains one or more copies of itself in case one of the copies becomes corrupt or is lost because of a hardware failure. Collectively, these sets of redo logs are referred to as *redo log groups*. Each multiplexed file within the group is called a *redo log group member*. Oracle automatically writes to all members of the redo log group to keep the files in sync. Each redo log group must be composed of one or more members. Each database must have a minimum of two redo log groups because redo logs are used in a circular fashion.

V\$LOG dynamic performance view shows information on redo logs in the database, their size along with other information. You can use the V\$LOGFILE dynamic performance view to view the names of the redo log groups and the names and locations of their members, as shown here:

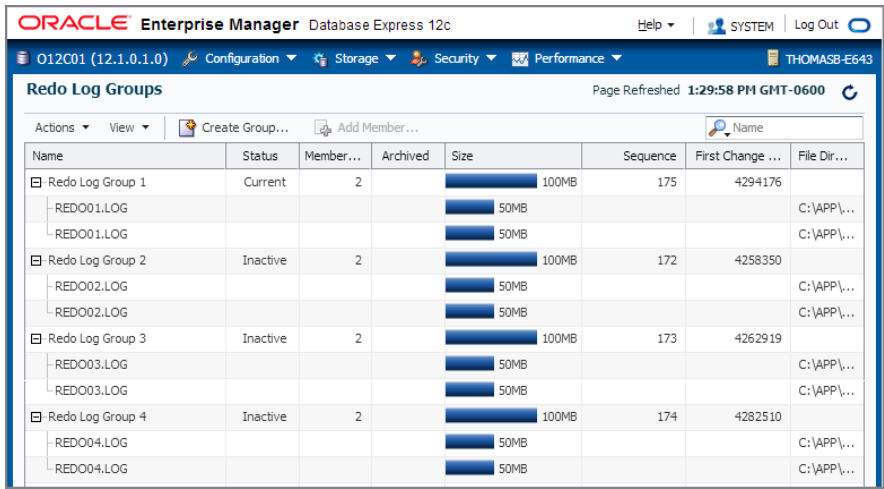
```
SQL> SELECT group#, member
       2 FROM v$logfile
       3* ORDER BY group#
SQL> /
```

```
GROUP# MEMBER
-----
1 C:\APP\ORACLE12C\MULTIPLEX\012C01\REDO01.LOG
1 D:\APP\ORACLE12C\ORADATA\012C01\REDO01.LOG
2 C:\APP\ORACLE12C\MULTIPLEX\012C01\REDO02.LOG
2 D:\APP\ORACLE12C\ORADATA\012C01\REDO02.LOG
3 C:\APP\ORACLE12C\MULTIPLEX\012C01\REDO03.LOG
3 D:\APP\ORACLE12C\ORADATA\012C01\REDO03.LOG
4 C:\APP\ORACLE12C\MULTIPLEX\012C01\REDO04.LOG
4 D:\APP\ORACLE12C\ORADATA\012C01\REDO04.LOG
```

```
8 rows selected.
SQL>
```

This output shows that the database has a total of four redo log groups and that each group has two members. Each of the members is located in a separate directory on the server's disk drives so that the loss of a single disk drive will not result in the loss of the recovery information stored in the redo logs. You can also monitor redo logs using EM Database Express, as shown in Figure 8.14.

FIGURE 8.14 EM Database Express showing redo logs



When a user performs a DML activity on the database, the recovery information for this transaction is written to the redo log buffer by the user’s server process. LGWR eventually writes this recovery information to the active redo log group until that log group is filled. Once the current log fills with transaction information, LGWR switches to the next redo log until that log group fills with transaction information, and so on, until all available redo logs are used. When the last redo log is used, LGWR wraps around and starts using the first redo log again. As shown in the following query, you can use the V\$LOG dynamic performance view to display which redo log group is currently active and being written to by LGWR:

```
SQL> SELECT group#, members, status
2 FROM v$log
3 ORDER BY group#;
```

GROUP#	MEMBERS	STATUS
1	2	CURRENT
2	2	INACTIVE
3	2	INACTIVE
4	2	ACTIVE

This output shows that redo log group number 1 is current and being written to by LGWR. Once redo log group 4 is full, LGWR switches back to redo log group 1. The following are the statuses available for log files.

- UNUSED - Online redo log is new and never been written to.
- CURRENT - The current active redo log.

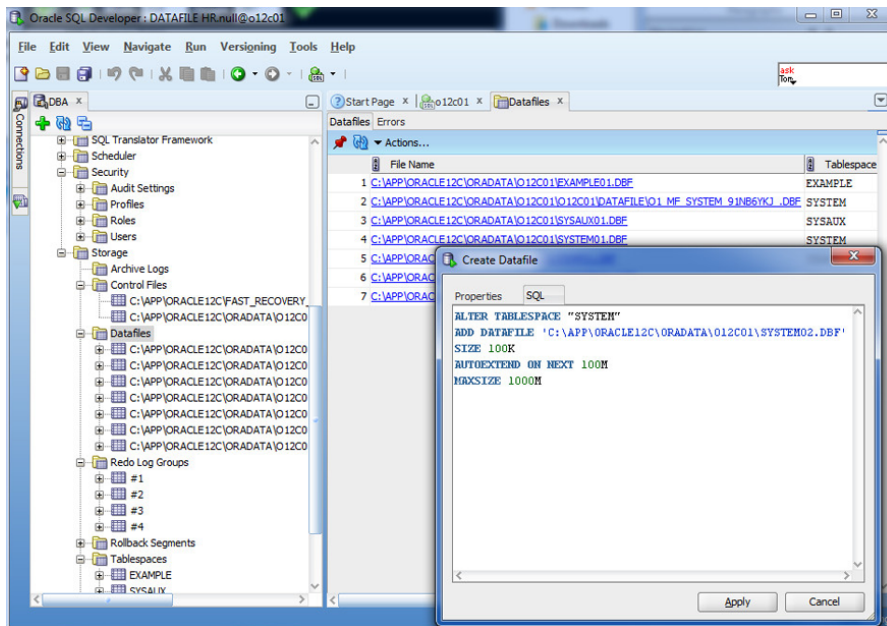
- **ACTIVE** - Log is active but is not the current log. It is needed for crash recovery.
- **CLEARING** - A short time status during `ALTER DATABASE CLEAR LOGFILE` statement. After the log is cleared, the status changes to **UNUSED**.
- **CLEARING_CURRENT** - Current log is being cleared of a closed thread. The log file may be in this status if there is an I/O error writing the new log information.
- **INACTIVE** - Log is no longer needed for instance recovery.

When LGWR wraps around from the last redo log group back to the first redo log group, any recovery information previously stored in the first redo log group is overwritten and, therefore, no longer available for recovery purposes.

However, if the database is operating in *archive log mode*, the contents of these previously used logs are copied to a secondary location before the log is reused by LGWR. If this archiving feature is enabled, it is the job of the `ARCn` background process described in the previous section to copy the contents of the redo log to the archive location. These copies of old redo log entries are called *archive logs*. Figure 8.15 shows this process graphically.

In Figure 8.15, the first redo log group has been filled, and LGWR has moved on to redo log group 2. As soon as LGWR switches from redo log group 1 to redo log group 2, the `ARCn` process starts copying the contents of redo log group 1 to the archive log file location. Once the first redo log group is safely archived, LGWR is free to wrap around and reuse the first redo log group once redo log group 3 is filled.

FIGURE 8.15 How `ARCn` copies redo log entries to disk





Nearly all production databases run in archive-log mode because they need to be able to redo all transactions since the last backup in the event of a hardware failure or user error that damages the database.

A database can have multiple archive processes and multiple archive destinations. We will discuss archiving and how the archived redo logs are used for database recovery in Chapter 15.



If LGWR needs to write to the redo log group that ARC*n* is trying to copy but cannot because the destination is full, the database hangs until space is cleared on the drive.

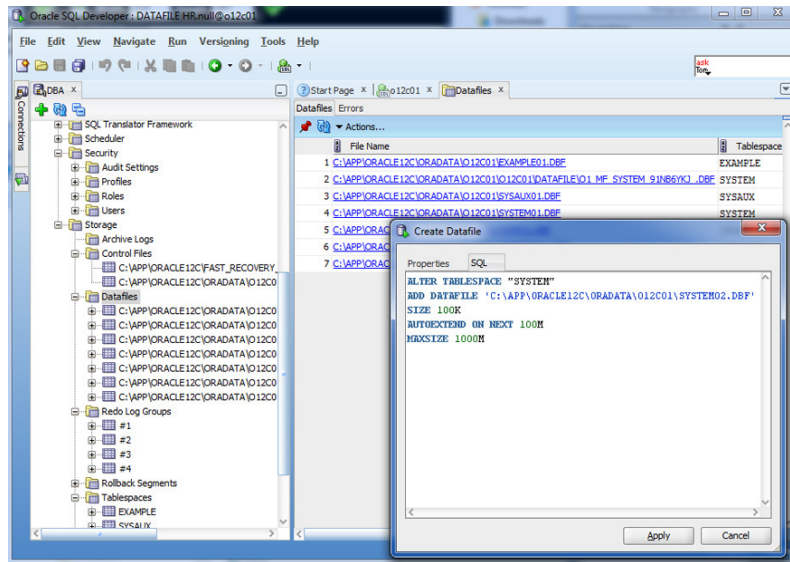
SQL Developer also has friendly menu options to manage database storage structures easily for DBAs. Figure 8.16 shows the storage menu screen from SQL Developer, giving you an overview. We encourage you to go through the menu items, modify the configuration, and use the SQL tab to view the SQL code generated by the SQL Developer tool. This will help you obtain a good understanding of the options and the syntax.

Real Application Clusters Database

Oracle *Real Application Clusters* were introduced in Oracle9i and have seen major enhancements in the Oracle 10g database where “grid” is the key. In the RAC architecture, there is one storage structure (database), with multiple Oracle instances (memory and processes) running on multiple nodes. This architecture gives high availability and horizontal scalability. When you need more capacity, all you need to do is add one more node to the RAC cluster.

In the RAC architecture, certain components must be shared by the instances; some can be shared by the instances and some components cannot be shared.

- Control files belong to the database, and all instances use the same control files.
- Database files belong to the database, and all instances access the same data files and permanent and temporary tablespaces.
- Each instance undo is kept separately and, therefore, requires that each instance undo its tablespace separately.
- Each instance has its own redo log buffer and redo threads and, therefore, has its own redo log files. Redo log files are not shared by the instances, but the files must reside in a shared location for recovery and backup purposes.
- It is advisable to keep the parameter file in a shared location accessible to all instances, with instance-specific parameters prefixed with the instance name.

FIGURE 8.16 The SQL Developer screen showing database storage

The Logical Structure

In the previous section, you saw how the Oracle database is configured physically. The obvious question is where and how your table is stored in a database. Let's try to relate the physical storage to the logical structures you know, such as tables and indexes.

Oracle logically divides the database into smaller units to manage, store, and retrieve data efficiently. The following paragraphs give you an overview of the logical structures:

Tablespaces The database is logically divided into smaller units at the highest level, called *tablespaces*. A tablespace has a direct relationship to the physical structure—a data file can belong to one and only one tablespace. A tablespace could have more than one data file associated with it.

A tablespace commonly groups related logical structures together. For example, you might group data specific to an application in a tablespace. This will ease the management of the application from the DBA's point of view. This logical division helps administer a portion of the database without affecting the rest of it. Each Oracle Database 12c database must have at least three tablespaces: SYSTEM, SYSAUX, and TEMP. For better management and performance, it must have two more tablespaces holding the UNDO data and application data.

Tablespaces are discussed in detail in Chapter 10.

Blocks A *block* is the smallest unit of storage in Oracle. A block is usually a multiple of the operating-system block size. A data block corresponds to a specific number of bytes of storage space. The block size is based on the parameter DB_BLOCK_SIZE and is determined when the database is created.

Extents An *extent* is the next level of logical grouping. It is a grouping of contiguous blocks, allocated in one chunk. Because they are allocated in contiguous chunks, extents cannot spawn multiple data files.

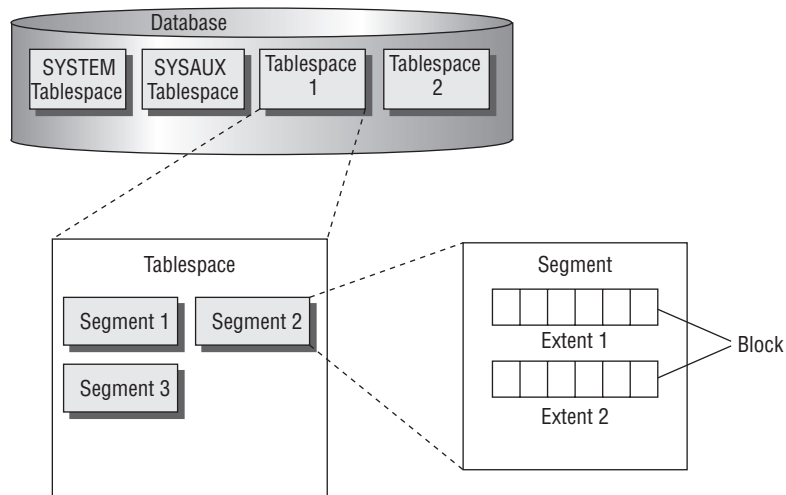
Segments A *segment* is a set of extents allocated for logical structures such as tables, indexes, clusters, table partitions, materialized views, and so on. Whenever you create a logical structure that stores data, Oracle allocates a segment, which contains at least one extent, which in turn has at least one block. A segment can be associated to only one tablespace; the extents of a segment may belong to more than one datafile. A segment is created when a table, index, materialized view, or a clustered table is created. When partitioned tables or partitioned indexes are created, one segment is created for each partition.

Figure 8.17 shows the relationship between data files, tablespaces, segments, extents, and blocks.

A schema is a logical structure that groups the database objects. A schema is not directly related to a tablespace or to any other logical storage structure. The objects that belong to a schema can reside in different tablespaces, and a tablespace can have objects that belong to multiple schemas. Schema objects include structures such as tables, indexes, synonyms, procedures, triggers, database links, and so on.

The DBA sees and manages the physical structure and logical structures of the database, whereas the programmer or a database user sees only the logical storage structures such as tables, indexes, and materialized views. They are not interested or not required to know to which tablespace the table belongs or where the tablespace data files are stored.

FIGURE 8.17 Logical database structure





Real World Scenario

Exploring the Data Dictionary for Physical and Logical Structures

The Oracle Data Dictionary is discussed in detail in Chapter 9. You may do this exercise after the database is created, but we wanted to document the information where it is relevant.

Here are a few data dictionary objects you can use to explore and help you understand the physical and logical structures more. You can use SQL*Plus or SQL Developer to explore. However, because SQL Developer shows a spreadsheet-like output, we recommend that you use SQL Developer for improved readability; just execute `SELECT * FROM <dictionary_view>` in the Worksheet.

Physical Storage Structures

Use the following v\$ and DBA views to explore your database's physical storage.

Control Files

V\$CONTROLFILE

Redo Log Files

V\$LOG

V\$LOGFILE

Data Files—The tablespace number (V\$) or name (DBA) links a data file to its logical storage structure

V\$DATAFILE

V\$TEMPFILE

DBA_DATA_FILES

DBA_TEMP_FILES

Logical Storage Structures

Use the following DBA views to explore the logical storage structures, and see how they are linked to physical storage. All of the views listed here have a file number column that ties the tablespace, segment, or extent back to its physical storage.

Tablespaces

DBA_TABLESPACES

Segments—The tablespace name links to DBA_TABLESPACES, segment's parent in the relational model for database logical structures.

DBA_SEGMENTS

Extents—The tablespace name ties extents to `DBA_TABLESPACES`, and the segment owner, segment type, and segment name combination ties each extent to a segment in the relational model.

`DBA_EXTENTS`

Summary

This chapter introduced you to the Oracle Database 12c architecture with the components that constitute an Oracle database server. Most popular databases today are relational databases. Relational databases consist of data composed of a set of relational objects. Data is stored in tables as rows and columns. Oracle is a relational database. SQL is the language used to manage and administer Oracle databases. Several tools are available to administer Oracle Database 12c. The most common ones used by DBAs are SQL*Plus and Oracle Enterprise Manager. SQL Developer is a GUI tool that can be used to interact with Oracle Database 12c using several DBA functions readily coded in menu items.

The Oracle Database 12c architecture consists of three major components: memory, processes, and storage. A user process initiates a connection with the Oracle database and starts a server process. The server process is responsible for performing the tasks on the database. The memory structures and background processes together are an Oracle instance. The server process communicates with the memory structure known as the system global area. The SGA consists of a shared pool, database buffer cache, and redo log buffer. The shared pool also includes components such as a Java pool, large pool, result cache, and streams pool.

There are many types of background processes, each performing a specific job to maintain and manage the database instance. All databases have at least nine background processes: the important ones are database writer, checkpoint writer, log writer, process monitor, and system monitor. Depending on the configuration of the database, there may be other background processes such as archiver, ASM balancing, and so on.

The physical data structure consists of several files stored on disk. The most important file is the control file, which keeps track of several important pieces of information, such as database name, names of data files and redo log files, backup information, and so on. The CKPT process is responsible for keeping the control file updated. Redo log files contain information from the redo log buffer. The LGWR process is responsible for writing the redo log buffer contents to the redo log files. Oracle metadata and application data are stored in data files. The DBWn process is responsible for writing dirty blocks from the database buffer cache to the data files.

Looking at the logical structure of the database, a tablespace is the highest level of logical unit. A tablespace consists of several segments. A segment consists of one or more extents. An extent is a contiguous allocation of blocks. A block is the smallest unit of storage in an Oracle database.

Exam Essentials

Describe common Oracle tools and their uses. Know which tools are available for connecting to and interacting with an Oracle database. Understand how these tools differ from one another.

Understand the Oracle architecture components. Be able to describe the logical and physical components of the Oracle architecture and the components that make up each. Know the relationship between segments, extents, database blocks, and operating-system blocks.

Understand the difference between a traditional Oracle database and a multitenancy database. Multitenancy databases are known as container databases and can have one or more pluggable databases. A traditional database is one database.

Know the background processes. Understand the Oracle Database 12c background processes and how they are used. The important ones to know are DBWn, CKPT, LGWR, PMON, SMON, ARCn, ASMB, RBAL.

Identify the three types of database files that constitute the database. Understand the purposes and key differences between the control files, data files, and redo log files.

Explain and categorize the SGA memory structures. Identify the SGA areas along with the subcomponents contained within each of these areas.

Review Questions

1. Choose two SGA structures that are required in every Oracle instance.
 - A. Large pool
 - B. Shared pool
 - C. Buffer cache
 - D. Java pool
2. Which statement is true?
 - A. A database can have only one control file.
 - B. A database must have at least two control files.
 - C. A database may have zero or more control files.
 - D. A database must have at least one control file.
3. Which component is configured at database startup and cannot be dynamically managed?
 - A. Redo log buffer
 - B. Streams pool
 - C. Java pool
 - D. Shared pool
 - E. None of the above
4. Which component is not part of an Oracle instance?
 - A. System global area
 - B. Process monitor
 - C. Control file
 - D. Shared pool
 - E. None
5. Which background process guarantees that committed data is saved even when the changes have not been recorded in data files?
 - A. DBWn
 - B. PMON
 - C. LGWR
 - D. CKPT
 - E. ARCn

6. User John has updated several rows in a table and issued a commit. What does the DBW_n (database writer) process do at this time in response to the commit event?
 - A. Writes the changed blocks to data files.
 - B. Writes the changed blocks to redo log files.
 - C. Triggers checkpoint and thus LGWR writes the changes to redo log files.
 - D. Does nothing.
7. Which of the following best describes a RAC configuration?
 - A. One database, multiple instances
 - B. One instance, multiple databases
 - C. Multiple databases plugged in from multiple servers
 - D. Multiple databases, multiple instances
8. Which component of the SGA contains the parsed SQL code?
 - A. Database buffer cache
 - B. Dictionary cache
 - C. Library cache
 - D. Parse cache
9. Which tasks are accomplished by the SMON process? (Choose all that apply.)
 - A. Performs recovery at instance startup
 - B. Performs cleanup after a user session is terminated
 - C. Starts any server process that stopped running
 - D. Coalesces contiguous free space in dictionary-managed tablespaces
10. Choose the best statement from the options related to segments.
 - A. A contiguous set of blocks constitutes a segment.
 - B. A nonpartitioned table can have only one segment.
 - C. A segment can belong to more than one tablespace.
 - D. All of the above are true.
11. From the following list, choose two processes that are optional in an Oracle Database 12c database.
 - A. MMON
 - B. MMNL
 - C. ARC_n
 - D. MMAN

12. Which SGA component will you increase or configure so that RMAN tape backups do not use memory from the shared pool?
- A. Java pool
 - B. Streams pool
 - C. Recovery pool
 - D. Large pool
13. When a user session is terminated, which processes are responsible for cleaning up and releasing locks? (Choose all that apply.)
- A. DBWn
 - B. LGWR
 - C. MMON
 - D. PMON
 - E. SMON
14. The LRU algorithm is used to manage what part of the Oracle architecture?
- A. Users who log on to the database infrequently and may be candidates for being dropped
 - B. The data file that stores the least amount of information and will need the least frequent backup
 - C. The tables that users rarely access so that they can be moved to a less active tablespace
 - D. The shared pool and database buffer cache portions of the SGA
15. Two structures make up an Oracle server: an instance and a database. Which of the following best describes the difference between an Oracle instance and a database?
- A. An instance consists of memory structures and processes, whereas a database is composed of physical files.
 - B. An instance is used only during database creation; after that, the database is all that is needed.
 - C. An instance is started whenever the demands on the database are high, but the database is used all the time.
 - D. An instance is configured using a pfile, whereas a database is configured using a spfile.

16. Which of the following is the proper order of Oracle's storage hierarchy, from smallest to largest?
- A. Operating-system block, database block, segment, extent
 - B. Operating-system block, database block, extent, segment
 - C. Segment, extent, database block, operating-system block
 - D. Segment, database block, extent, operating-system block
17. The DBA unknowingly terminated the process ID belonging to the PMON process of Oracle Database 12c database using the `kill -9` command on Unix. Choose the best answer:
- A. Oracle spawns another PMON process automatically.
 - B. The database hangs, and the DBA must manually start a PMON process.
 - C. If the database is in ARCHIVELOG mode, Oracle automatically starts another PMON process and recovers from the database hang.
 - D. The instance crashes and needs to be restarted.
18. When an incremental checkpoint happens in a database, which file(s) are updated with the checkpoint position? Choose all options that are correct.
- A. Data files
 - B. Control files
 - C. Initialization Parameter Files
 - D. Redo log files
 - E. Archive log files
19. User Isabella updates a table and commits the change after a few seconds. Which of the following actions are happening in the database? Order them in the correct sequence and ignore the actions that are not relevant.
- A. Oracle reads the blocks from data file to buffer cache and updates the blocks.
 - B. Changed blocks from the buffer cache are written to data files.
 - C. The user commits the change.
 - D. LGWR writes the changed blocks to the redo log buffer.
 - E. The server process writes the change vectors to the redo log buffer.
 - F. LGWR flushes the redo log buffer to redo log files.
 - G. A checkpoint occurs.

- 20.** Querying the V\$LOG file shows the following information. Which redo group files are required for instance crash recovery?

```
SQL> select GROUP#, ARCHIVED, STATUS from V$LOG;
```

```
GROUP# ARC STATUS
----- --
1 NO CURRENT
2 NO INACTIVE
3 NO INACTIVE
4 NO ACTIVE
```

- A.** Group 1 and 4
- B.** Group 2 and 3
- C.** Groups 1 through 4
- D.** Group 1
- E.** Group 4