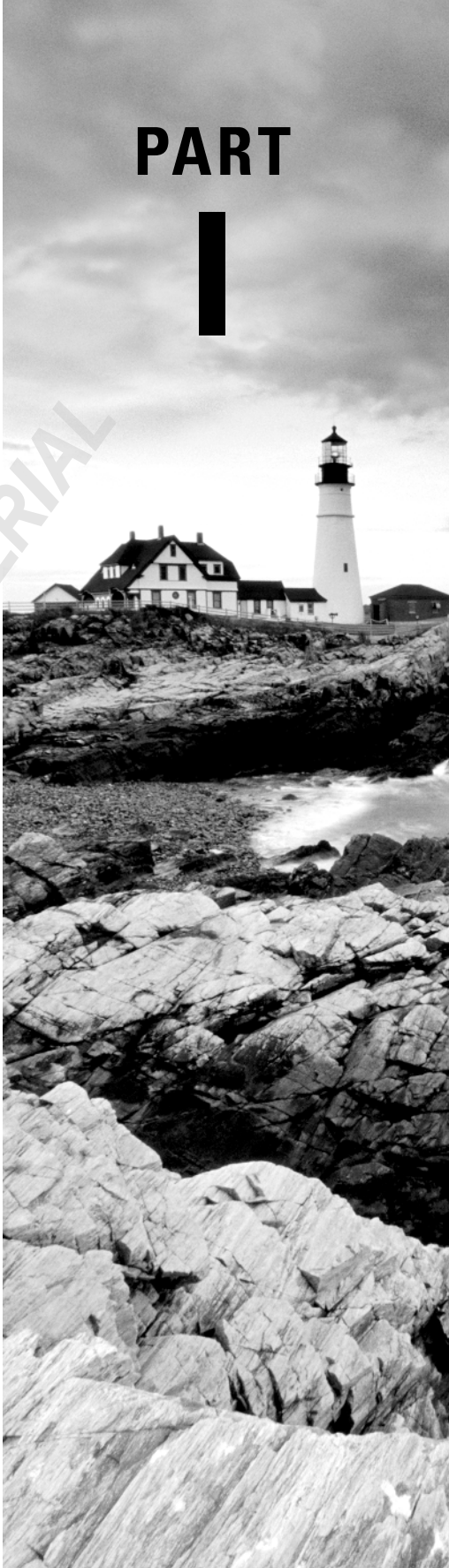
Oracle Database 12*c*: SQL Fundamentals

PART I



Chapter 1

Introducing Oracle Database 12*c* RDBMS

ORACLE DATABASE 12*c*: SQL FUNDAMENTALS EXAM OBJECTIVES COVERED IN THIS CHAPTER:

✓ **Introduction**

- Describe the features of Oracle Database 12*c*.
- Describe the salient features of Oracle Cloud 12*c*.
- Explain the theoretical and physical aspects of a relational database.
- Describe Oracle server's implementation of RDBMS and object relational database management system (ORDBMS).

Organizations and individuals collect and use a variety of information (data). A database collects data, stores and organizes data, and retrieves related data used by a business. Oracle is the world's most widely used database management system. With the release of its Database 12*c*, Oracle has enhanced the capabilities of its feature-rich database to include cloud architecture. The *c* in 12*c* stands for cloud computing. From Oracle version 8 onward, Oracle includes the core emphasis of the release along with the version number in its name. Versions 8 and 9 are called *i* to indicate Internet computing; versions 10 and 11 are called *g* for grid computing.

With the cloud enablement, Oracle Database 12*c* lets you manage many databases as one, thereby reducing overhead and valuable resource consumption.

This chapter will introduce you to the Oracle Database 12*c* high-level components and how the Oracle database is organized. You will also learn about the relational and object capabilities of the database, and the tools available for database administrators (DBAs) to retrieve information and manage the database.

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://www.oracle.com/education/certification` for the most current exam objectives.

Relational Database Management Systems

A *database management system* (DBMS) controls the storage, organization, and retrieval of data. In a DBMS, the kernel code is the software piece that manages the storage and memory component of the database. There is metadata in the DBMS that keeps track of all the components of the database, also known as the dictionary. The code or language used to retrieve data from the database is known as SQL, which stands for *Structured Query Language*.

Over the years, database management systems have evolved from hierarchical to network to relational database management systems (RDBMS). A *relational database management system* is an organized model of subjects and characteristics that have relationships among the subjects. A well-designed relational database provides volumes of information about a business or process. *RDBMS* is the most widely used database system, and the object

structures are related. We see relationships everywhere in our daily lives: parents and children, team and players, doctor and patient, to name a few. The main advantages of RDBMS include the way it stores and retrieves information and how the data integrity is maintained. RDBMS structures are easy to understand and build. These structures are logically represented using the *entity-relationship (ER) model*. The exam will have one or two questions on the ER diagram and/or the RDBMS concept. You may already be familiar with the RDBMS concepts and ER diagrams; a brief refresher is included here.

Characteristics of a Relational Database

Relational databases have the following three major characteristics that constitute a well-defined RDBMS:

- **Structures** are objects that store or access data from the database. Tables, views, and indexes are examples of structures in Oracle.
- **Operations** are the actions that are used to define the structures or to manipulate data between the structures. `SELECT` statements and `CREATE` statements are examples of operations in Oracle.
- **Integrity rules** govern what kinds of actions are allowed on data and the database structure. These rules protect the data and the structure of the database. The primary keys and foreign keys are examples of integrity rules in Oracle.

Logical Model

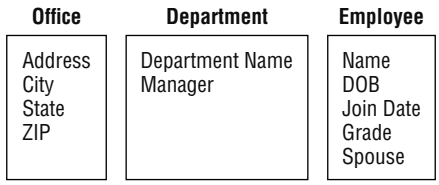
In the design phase of the system development cycle, a logical model of the database is created. A logical model of an RDBMS is typically a block diagram of entities and relationships, referred to as an *entity-relationship (ER) model* or ER diagram.

An ER model has *entity*, *relationship*, and *attributes*. An ER model is visual, showing the structure, characteristics, and interactions within and around the data being modeled.

Entities and Attributes An entity in a logical model is much like a noun in grammar—a person, place, or thing. The characteristics of an entity are known as its attributes. Attributes are detailed information about an entity that serves to qualify, identify, classify, or quantify it. For example: ABC Inc. has many offices in the United States; each office has many departments, and each department may have many employees. Placing the organization of ABC Inc. in terms of the ER model, you could identify OFFICE, DEPARTMENT, and EMPLOYEE as entities. Each entity will also have its own characteristics. For instance, when you say "office," you might want to know the address and city where the office is located, the state, and how many employees work there. Similarly, you might want to know the department name, its manager, the employee's name, date of birth, hiring date, and salary grade. You might also like to know the employee's spouse's name. See Figure 1.1.

There are optional and mandatory attributes. In Figure 1.1, the spouse's name, along with the employee information, is optional; whereas the employee name, the department he/she belongs to, hire date, and date of birth are mandatory in Figure 1.2. An asterisk along with the attribute name indicates that it is mandatory. The optional attribute may be indicated with a small *o*.

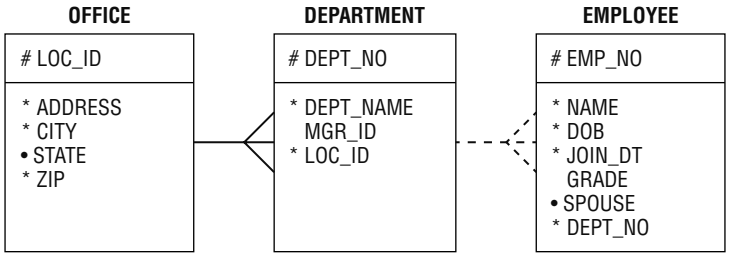
FIGURE 1.1 Entities and attributes



Relationships and Unique Identifiers In the example of ABC Inc., the relationship between the entities is described as "each office has many departments," "one department belongs to only one office," "each department has many employees," and "one employee can belong to only one department." If there is an office in one city, there should be at least one department. So it is mandatory to have at least one occurrence of department for each location. There may be many departments in one location. In the ER model, a solid line represents a mandatory relationship, and a crowfoot represents the "many." But in some departments, there may not be any employees at all. Optional occurrence is represented by a dotted line.

You should be able to identify each occurrence of an entity uniquely. Now what happens if there are two employees with the same name? How do you distinguish them? For office location, the city and state uniquely identify each office; for department, the department name identifies it uniquely. For employee, you can introduce a unique identifier (UID) called employee number. Figure 1.2 is a refined version of Figure 1.1, and it shows the entities, attributes, relationships, optional and mandatory attributes, and UIDs. UID is represented in the diagram using a pound (#) symbol.

FIGURE 1.2 An entity-relationship (ER) model



Three types of relationships can be defined between the entities. (Figure 1.3):

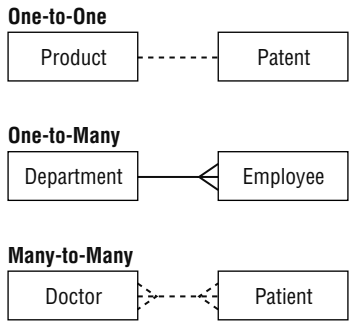
One-to-One A one-to-one relationship is one in which each occurrence of one entity is represented by a single occurrence in another entity. For example, product and patent—one product might have a patent, and one patent corresponds to only one product.

One-to-Many A one-to-many relationship is one in which an occurrence of one entity can be represented by many occurrences in another entity. For example, department and

employees—one department has one or more employees, and an employee belongs to only one department.

Many-to-Many A many-to-many relationship is one in which an occurrence from one entity can be represented by one or more occurences in another entity, and an occurrence from the second entity may be represented by one or many occurences in the first entity. Many-to-many relationships should not exist in RDBMS because they cannot be properly joined to represent a single row correctly. To solve this, create another entity that has an one-to-many relationship with the first entity and an one-to-many relationship with the second entity. For example, doctor and patient—a patient can visit many doctors, and a doctor can have many patients.

FIGURE 1.3 Types of relationships



The logical model also provides information known as *access paths*. They are the common ways you usually query the database in order to retreive information. For example, you would always query the employee records with the Dept_No or Emp_No. Think of the access paths as an index to the data; they help us locate data just as the index of a book helps us quickly find the information we need.

When you have established the relationships between entities, it's time to normalize the design. *Normalization* is the process of eliminating redundant information from the entities until you can uniquely identify each occurrence of the entity. This may not always be practical due to performance and implementation issues. In such cases, you can denormalize to some extent.

Physical Model

The physical model is created by taking the logical model and creating a database and database objects to represent the entities and relationships. In the physical model, each entity becomes a *table* and attributes of the entity become *columns* of the table. The relationship between the entities is part of one or more *constraints* between the tables. Physical implementations might force you to combine, separate, or create completely new entities in order to best realize the logical model. The unique identifiers of an entity become the *primary key* of the table. Stored procedures, functions, and triggers may be created to enforce business rules.

In RDBMS, the physical database storage is independent of the logical model.

Oracle's Implementation of RDBMS and ORDBMS

A database server is the key to information management. An Oracle database satisfies all three major characteristics of the relational model. Oracle lets you define tables, columns, column characteristics such as datatype, length, whether the values are mandatory, and default values. Defining *foreign key* ensures the *referential integrity* of the data. You can define *primary keys* and indexes on the data. The primary key of a relational table uniquely identifies each record in the table; it may consist of a single attribute (column) or multiple attributes in combination. A foreign key is a column (or collection of columns) in one table that uniquely identifies a row of another table, defining the relationship between the tables.

Records in a database table can be seen as instances of the entity. Each occurrence of an entity is differentiated by the values of the attributes. Oracle stores these records as *rows* of the table and the attributes as *columns* in each row. In the most generic form, a database table can be seen as a single spreadsheet with unlimited numbers of columns and rows. The columns are not defined until the user names them and gives them a datatype. Oracle extends the concept of spreadsheets by defining relationships between multiple spreadsheets, constraints on columns, and providing mechanisms for multiple users to access the same database table(s) at the same time.

The data access path is implemented in Oracle using indexes. Indexing allows us to predefine to the relational database system the most common access paths that will be used. These indexes decrease the time required to search for data in a table using a number of algorithms such as B-tree, bitmap, etc.

Oracle implements the RDBMS characteristics using the following set of structures:

- Tables are used for data storage.
- Views and synonyms are created for data access.
- Indexes are used to speed up data retrieval.
- Primary keys, foreign keys, and unique keys are called *constraints* and are created to enforce data integrity.
- Triggers are created to satisfy the business rules.
- Roles and privileges are used for security.
- Procedures, functions, and packages are used to code the application.

Oracle, since version 8i, is also an Object Relational DBMS. An RDBMS that implements object-oriented features such as user-defined types, inheritance, and polymorphism is called ORDBMS. It lets you create user-defined object types in the relational database system. Object types are structures that consist of built-in or user-defined data types. For example, `Address` can be defined as an object type and can be referenced in tables.

Here's an example where STREET_TYPE is defined as:

```
STREET_TYPE
STREET_NUMBER NUMBER   (6)
STREET_NAME1  VARCHAR2 (40)
STREET_SUFFIX VARCHAR2 (10)
APARTMENT_NO  VARCHAR2 (5)
```

Here's an example where ADDRESS_TYPE is an object type defined using another object type as:

```
ADDRESS_TYPE
STREET        STREET_TYPE
CITY          VARCHAR2 (30)
STATE         CHAR (2)
ZIP           NUMBER (5)
```

In this example for CUSTOMER_TABLE, the object CUST_ADDR is a type.

```
CUSTOMER_TABLE
CUST_NAME     VARCHAR2 (40)
CUST_ADDR     ADDRESS_TYPE
CUST_PHONE    VARCHAR2 (12)
CUST_FAX      VARCHAR2 (12)
```

Now that the ADDRESS_TYPE is defined, it can be used in any number of tables, where ADDRESS needs to be stored. This is a small example to show you how objects can be reused and how the functionality of the RDBMS can be extended to include built-in complex business rules.

The Oracle Database 12*c*

An Oracle Database 12*c* server is a feature-rich RDBMS that extends its capabilities beyond any other RDBMS in the market, with object relational and cloud capabilities. In this section, we will discuss the capabilities and features of Oracle Database 12*c*.

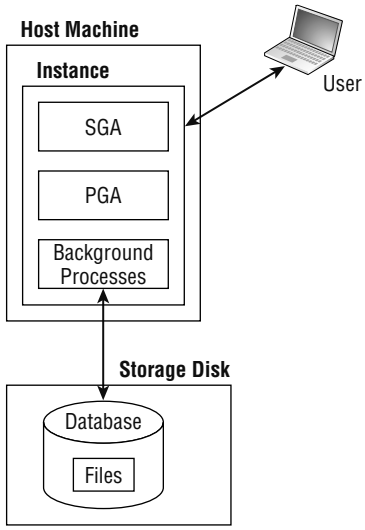
Oracle Database 12*c* Implementations

Let's start with the architecture of the database server at a very high level. Detailed architecture and components are discussed in various chapters in Part II of this book.

The physical structure of an Oracle Database 12*c* server consists of two major components: the *database* and the *instance*. The database is a set of physical files saved on the disk that store information. The instance is a set of memory structures and processes that uses the physical components to manipulate and retrieve data.

Figure 1.4 shows the database architecture. The host machine is where the Oracle instance is running. It has the memory structures and processes. The storage array, or disk, is where the database resides.

FIGURE 1.4 An Oracle database server



In the architecture shown in Figure 1.4, one instance communicates with one database. The host machine is where users and applications connect and interact. If the machine goes down for some reason, the database will be unavailable. Oracle alleviates this issue by introducing an architecture named the *Real Application Clusters* (RAC).

Figure 1.5 shows RAC architecture. In this architecture, more than one instance communicates to a single database. Oracle RAC takes reliability a step further by removing the database server as a single point of failure. If an instance fails, the remaining instances in the RAC pool remain open and active; and connections from failed instances can be failed-over to active instances. The RAC load balancer directs the user connection request to the appropriate instance.

With RAC, high availability and CPU/memory capacity available to the database is increased. Oracle manages the connection load balancing and failover automatically.

Many organizations have several hundreds or thousands of Oracle databases. Imagine if the policy were to have one instance per server, then you would have as many servers as the number of instances to manage. If you have a high-capacity server or if the database resource requirements are minimal, you can have more than one instance on the same host machine. Figure 1.6 shows an architecture in which more than one database is hosted on the same machine.

FIGURE 1.5 An Oracle database server — RAC

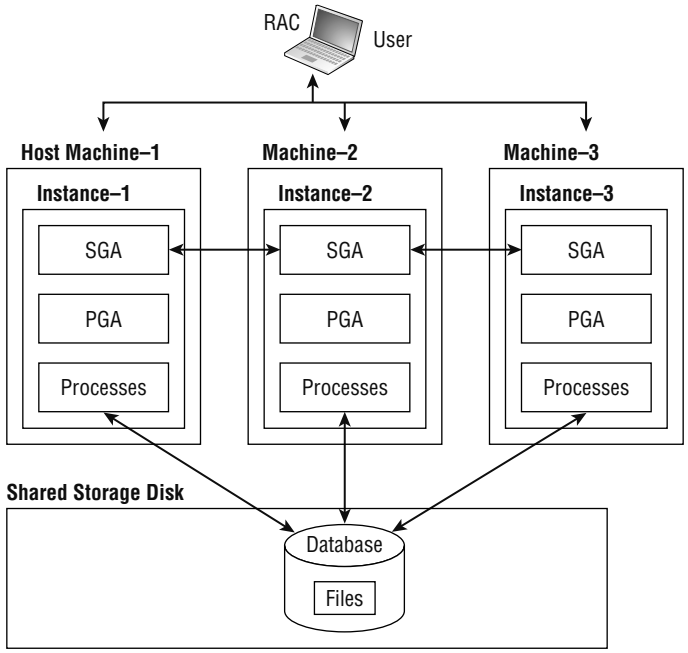


FIGURE 1.6 Multiple Oracle databases on same machine

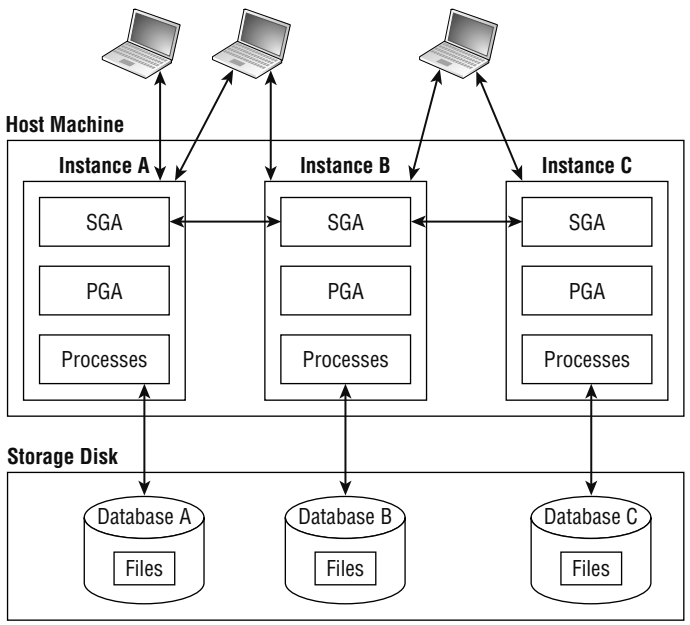
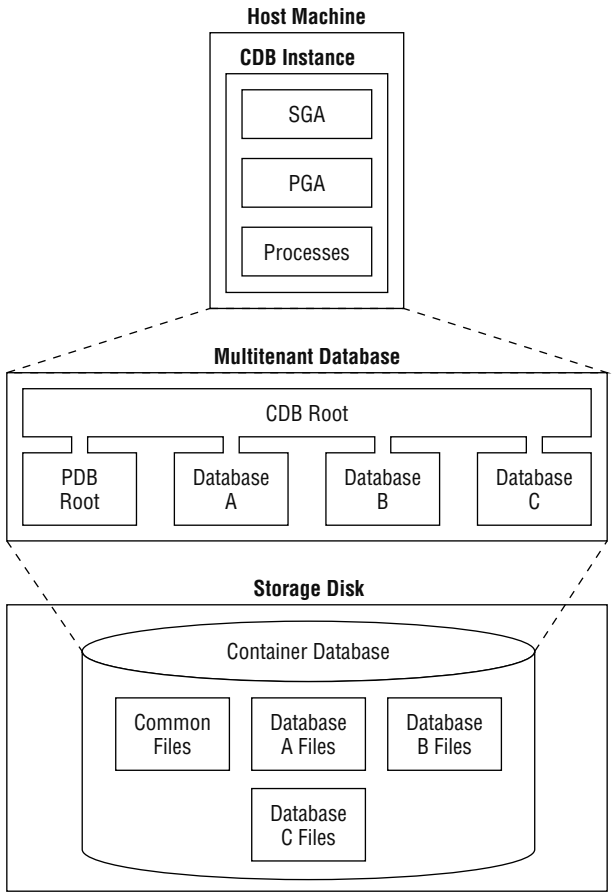


Figure 1.6 clearly shows that even though you consolidated multiple database servers to one host, you still have the same overhead of managing the database. You need instance memory structures, processes, and management activities such as backup for each instance or database.

With Oracle 12*c*, a new architecture feature is introduced known as the *multitenant architecture*. The multitenant architecture enables an Oracle database to function as a multitenant container database (CDB) that includes zero, one, or many *pluggable databases* (PDBs). All databases created prior to Oracle Database 12*c* are non-CDB; a pluggable database appears as a non-CDB to the application, so existing code and application need not be changed when you move to Oracle Database 12*c*.

The PDBs belonging to a CDB share the database overhead such as redo, undo, and memory. Oracle RDBMS is responsible for keeping the pluggable databases separate, private for the application, and secure. The instance and SGA are assigned to a container database. Figure 1.7 shows the *multitenant database* architecture. The databases that are part of the CDB are known as pluggable databases.

FIGURE 1.7 Oracle Database 12*c* — Multitenant architecture with pluggable databases



Pluggable databases in the multitenant architecture, as the name suggests, can be unplugged and plugged to another CDB easily. With the pluggable databases and multitenant architecture, Oracle Database 12*c* offers the following benefits:

- **Increased server utilization:** Because the overhead associated with each database is now shared among all databases, you can consolidate more databases with the same resources.
- **Cost reduction:** By consolidating hardware and sharing database memory and files, the cost of hardware is less.
- **Application transparency:** Although the architecture changed, each PDB acts and works as a traditional pre-12*c* Oracle database. There is no need to change application code or architecture to start using Oracle Database 12*c*.
- **Manage many databases as one:** Administrative activities such as patching and upgrade are performed on the container database so they do not need to be repeated for each database in the CDB. This drastically reduces the administrative time required.
- **Less backup configuration:** With multiple databases consolidated into one server, you still have to back up each database separately. With container and pluggable databases, you only need to back up one multitenant container database.
- **Easier provisioning:** In the container database architecture, it is very easy to clone and provision pluggable databases.
- **Less time to upgrade:** When you upgrade the container database, all the pluggable databases are automatically upgraded.
- **Move databases:** It is also possible to move a pluggable database from one container database to another. This is especially useful if you plan to upgrade all but a few databases to the next release. Before upgrading the container database, you may move a few pluggable databases to another container database, or move pluggable databases to another CDB of higher release.
- **Separation of duties:** Database administrators can be defined as a CDB administrator (common user with administrative privileges on all CDB and PDBs) or a PDB administrator (local user in PDB with administrative privilege only on the PDB).

In the next section, you will learn about users and schema in the database, which is the basis for connecting to the database.

Connecting to Oracle Database

Before you can connect to an Oracle database, you must create a user. When you create a new Oracle database, several default users are automatically created. (The preferred method for creating a database is to use the Database Configuration Assistant [DBCA], which is discussed in Chapter 9, "Creating and Operating Oracle Database 12*c*.")

`SYS` is the data dictionary or metadata dictionary owner for the database. Using this account for day-to-day operations is not recommended. You connect to `SYS` to start and stop database and other key administrative tasks. `SYSTEM` is a powerful administrative user in the database. Initially, you use this user connection to create database users and other

administrators in the database. Along with SYS and SYSTEM, several other database users are also created based on the options you choose during the installation and on the components installed in the database.

In a container database, the users are either common or local. Common users are visible on the container as well as in all pluggable databases; they have the same username and password across all pluggable databases and in the container database. The schema for common user is still local to each pluggable database and the container database.

A *schema* is a collection of database objects owned by a user account in the database. The objects in the schema may be related to support a business application. A schema and user have a one-to-one relationship in a database. A schema is created as a user in the database, but when the user owns database objects, it is called a schema. Schema objects are discussed in Chapter 7, "Creating Tables and Constraints."

When an object is created under someone's schema, the user has full privilege on the object by default. Using Oracle's roles and privileges, a schema owner or administrator can grant privilege on his or her object (such as a table) to another user in the same database. This is known as object-level privilege. For certain users, you may want to grant privileges on all the objects in the database; this is accomplished by using the system privileges. Privileges and system security are discussed in Chapter 13, "Implementing Security and Auditing."

The next section will introduce you to the tools available to manage and administer Oracle Database 12*c*.

Database Management Tools

Oracle Database 12*c* comes with multiple feature-rich tools to help administrators manage and monitor the database and all of its components. In this section, you will review the tools that are used for everyday administration of Oracle database. Let's start with the tool that is equally beneficial to DBAs, developers, and power users.

SQL Developer

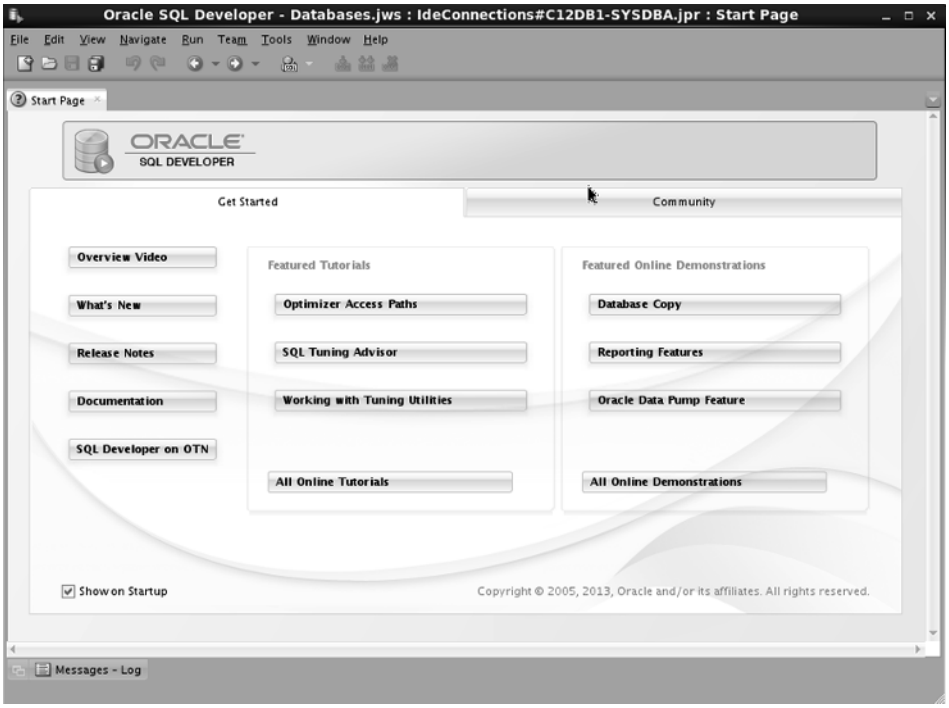
SQL Developer is a graphical tool used to perform everyday activities in the Oracle database. It has several predefined menu functions; therefore, there is no need to remember the syntax or SQL command to perform basic functions. In addition to Oracle (database versions higher than 9i Release 2), SQL Developer can also connect to Microsoft Access, Microsoft SQL Server, MySQL, IBM DB2, and Sybase Adaptive Server databases to view data and metadata.

SQL Developer is installed along with Oracle Database 12*c* on Windows platforms. For Linux, you must download and install it outside of an Oracle Database 12*c* installation. The distribution usually included in the database software distribution might not be the current version. It is better to always download the latest version and install SQL Developer from `http://www.oracle.com/technetwork/developer-tools/sql-developer/downloads`. As instructed on the download page, you will also need Java JRE installed for SQL Developer to work.

Once it is installed, you can invoke SQL Developer from `/usr/local/bin` directory on Linux or from the Oracle Installation program group on Windows. Figure 1.8 shows the

initial SQL Developer screen. To get started and learn more about SQL Developer, click the choices available in the screen.

FIGURE 1.8 The initial SQL Developer screen

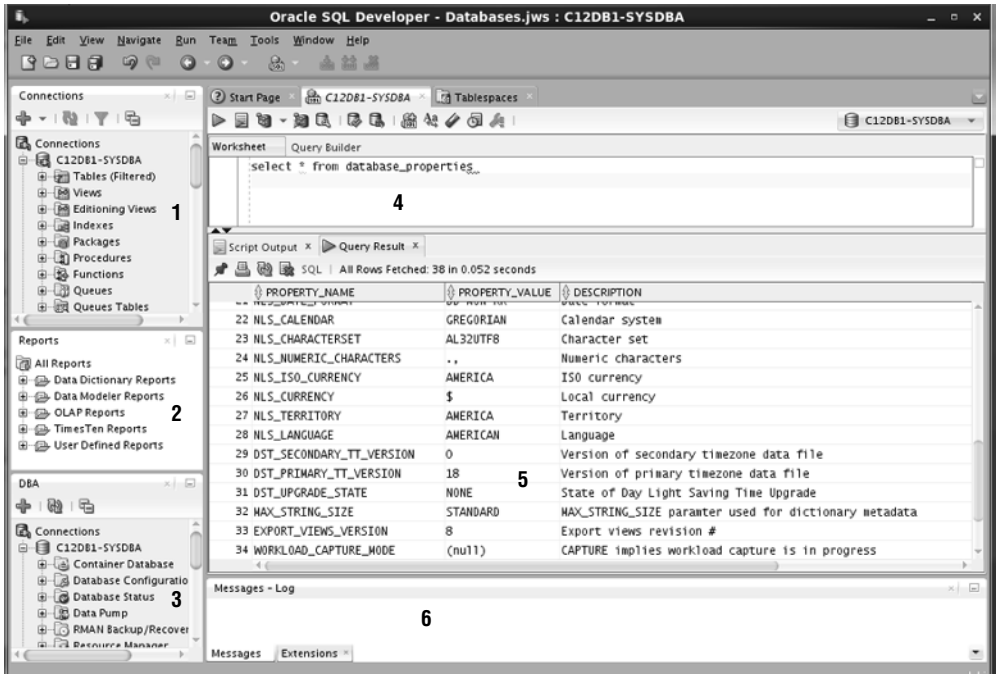


The Connections navigation pane is the most commonly used pane, which navigates through the database objects providing you with information and options to modify the objects. In Figure 1.9, you can see the various sections and navigations in SQL Developer. The section marked 1 shows various database connections and the objects in those databases. Section 2 provides you with a set of predefined reports. If you know how to write SQL, you can define your own reports as well. Section 3 shows the DBA navigation screen, showing you various DBA tasks.

Section 4 is the SQL worksheet. You use this section to interact directly with the database using SQL language. Output from the SQL commands is listed in section 5; both query output and script output are visible. Section 6 is the logging pane. This is useful when you are debugging code.

For simplicity and ease of results capture, the SQL statements used in this book are using SQL*Plus. All of the SQL commands can also run using the SQL Worksheet in SQL Developer.

FIGURE 1.9 SQL Developer navigation windows



SQL*Plus

*SQL*Plus* is Oracle's command-line interface to the Oracle database. You run SQL commands to query the database or to manage the database. SQL*Plus is packaged with the Oracle software and can be installed using the client software installation routine on any machine. This tool is automatically installed when you install the Oracle Database 12*c* server software.

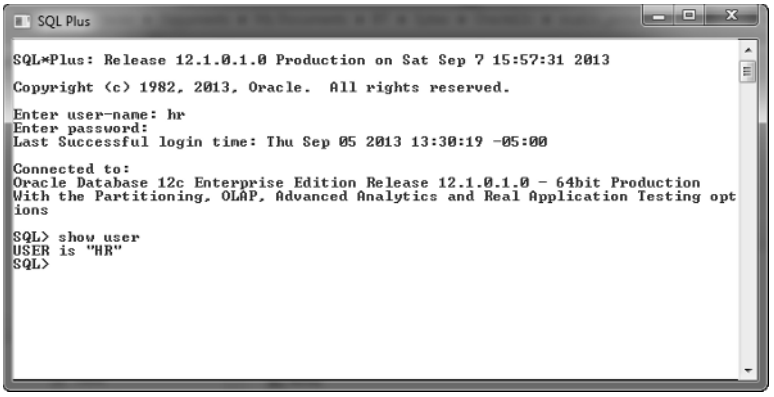
On Unix/Linux platforms, you can invoke SQL*Plus using the `sqlplus` executable found in the `$ORACLE_HOME/bin` directory. On Windows, SQL*Plus is under the Oracle Home Group menu. On Windows and Unix/Linux platforms, when you start SQL*Plus, you will be prompted for a username and password, as shown in Figure 1.10.

Once you are in SQL*Plus, you can connect to another database or change your connection by using the `CONNECT` command, with this syntax:

```
CONNECT <username>/<password>@<connect_string>
```

The slash separates the username and password. The connect string following `@` is the database alias name known as the net service name. If you omit the password, you will be prompted to enter it. If you omit the connect string, SQL*Plus tries to connect you to the local database defined in the `ORACLE_SID` variable (or the net service name defined by `TWO_TASK` variable). You will not be prompted for the `<connect_string>`. Include `<connect_string>` along with `<username>` as in `bthomas@myowndb`.

FIGURE 1.10 A SQL*Plus screen



You may replace the connect string with a construct called the easy connect. The syntax is `[//]`*Host*`[:`*Port*`]/<`*service_name*`>]`. For example, to connect to database service named C12DB1 on machine BTLNX63, where the listener is running in port 1521, use

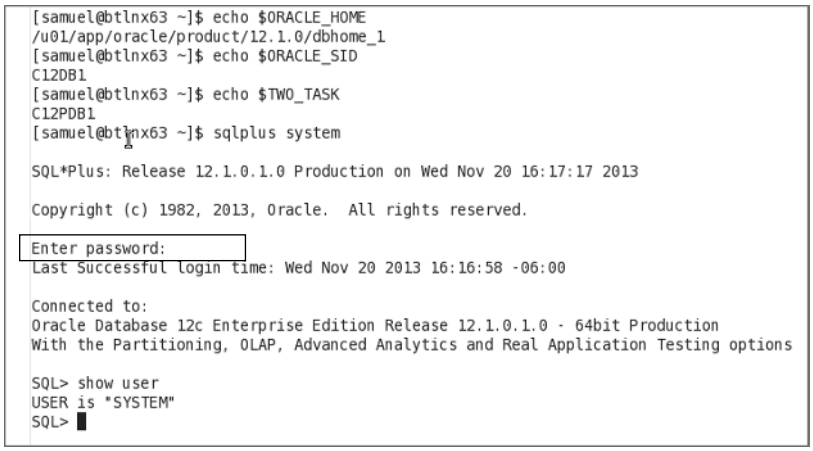
```
sqlplus system@"btlnx63:1521/C12DB1"
```

You can invoke and connect to SQL*Plus using the `sqlplus` command, with this syntax:

```
sqlplus <username>/<password>@<connectstring>
```

If you invoke the tool with just `sqlplus`, you will be prompted for a username and password, as in Figure 1.10. If you invoke SQL*Plus with a username, you will be prompted for a password. See Figure 1.11 for an example.

FIGURE 1.11 A SQL*Plus screen with a password prompt



Once you are connected to SQL*Plus, you will get the `SQL>` prompt. This is the default prompt, which can be changed using the `SET SQLPROMPT` command. Type the command you want to execute at this prompt. With SQL*Plus, you can enter, edit, and execute SQL statements; perform database administration; and execute statements interactively by accepting user input. You can also format query results and perform calculations.

To exit from SQL*Plus, use the `EXIT` command. On platforms where a return code is used, you can provide a return code while exiting. You can also use the `QUIT` command to complete the session. `EXIT` and `QUIT` are synonymous.

The command `sqlplus -help` displays a help screen to show the various options available when starting SQL*Plus. Multiple administrative connections such as SYSDBA, SYSOPER, SYSBACKUP, SYSASM, SYSKM, and SYSDG are also available. They are discussed in this book in various chapters.

Oracle Enterprise Manager Database Express 12*c*

Oracle Enterprise Manger (OEM) Database Express 12*c* is a web-based tool that can be configured using default by Database Configuration Assistant (DBCA, the tool you use to create and configure databases) when you create an Oracle database. OEM Database Express by default uses port 5500, hence is invoked using URL `https://database_host_machine:5500/em`.

OEM Database Express is designed to manage only one database, and is intended for database administrators. When you invoke Database Express, you will be prompted for a username and password to connect to the database. You should provide a user account that has administrative privileges.

For the SQL section of this book, you will not be using this tool, so we will not discuss it in more detail here. You can read more information about OEM Database Express in Chapter 3, "Getting Started with Database Administration," of "Oracle Database 2 Day DBA 12*c* Release 1 (Part E17643-12)," found at `http://docs.oracle.com/`.

Oracle Enterprise Manager Cloud Control 12*c*

Oracle Enterprise Manager Cloud Control 12c (OEM 12c) is Oracle's integrated enterprise management administrative tool, providing complete cloud management solutions. With OEM 12*c*, you can manage multiple databases and all products under the Oracle stack. It is a complete cloud lifecycle management answer to quickly set up, administer, and support enterprise clouds and Oracle environments from applications to storage.

OEM 12*c* is not part of the Oracle Database 12*c* software install; it must be downloaded and installed separately. To read more about OEM 12*c*, please check out "Oracle Enterprise Manager Cloud Control Introduction 12*c* Release 3 (Part E25353-14)" at `http://docs.oracle.com/`.

Oracle Database 12*c* in the Cloud

Cloud architecture emphasizes sharing resources and maximizing the effectiveness of shared resources. Cloud resources are shared not only by multiple users, but are also capable of reallocation based on demand. Cloud computing allows organizations to provision resources and applications rapidly, with improved manageability and less administrative overhead.

By enabling customers to efficiently use their information technology infrastructure, Oracle Database 12*c* was designed for the cloud. The following are the benefits of having Oracle Database 12*c* in the cloud architecture:

- It consolidates multiple Oracle databases into multitenant container databases.
- With multitenant architecture, DBAs can manage multiple databases as one database for many administrative tasks on the database. DBAs need to perform fewer patches and upgrades and will not need to configure many backups.
- It automatically optimizes database storage and performance based on usage.
- Oracle Database 12*c* supports smart compression and storage tier. The heat map feature tracks data usage information; administrators can create appropriate policies to automatically move and compress data based on age and activity of data.
- Oracle RAC supports deployment of database instances across a pool of servers, helping to avoid downtime caused by unplanned server outages.
- With Oracle Enterprise Manager Cloud Control 12*c*, the provisioning and cloning of databases are simplified.

Oracle Database 12*c* helps customers reduce IT complexity and cost through private database cloud deployments by consolidation. Cloud computing offers an opportunity for IT organizations to be more responsive to changes in application workloads and business demands.

Because the test is on SQL and the tool used throughout the book for executing SQL is SQL*Plus, the next section will discuss some fundamentals of SQL*Plus.

Becoming Familiar with SQL*Plus

SQL*Plus, widely used by DBAs and developers to interact with a database, is a powerful tool from Oracle. Using SQL*Plus, you can execute all SQL statements and PL/SQL programs, format results from queries, and administer the database.

Earlier in this chapter, you learned how to connect to the database using SQL*Plus. In this section, you will learn about entering SQL commands, understanding the difference between SQL commands and SQL*Plus commands, editing the SQL*Plus buffer, and running commands in a script.

Entering SQL Statements

A SQL statement can spread across multiple lines, and the commands are not case sensitive. The previously executed SQL statement will always be available in the *SQL buffer*.

The buffer can be edited or saved to a file. You can terminate a SQL statement in any of the following ways:

- End with a semicolon (;): The statement is completed and executed.
- Enter a slash (/) on a new line by itself: The statement in the buffer is executed.
- Enter a blank line: The statement is saved in the buffer.

You can use the `RUN` command instead of a slash to execute a statement in the buffer. The SQL prompt returns when the statement has completed execution. You can enter your next command at the prompt.

Only SQL statements and PL/SQL blocks are stored in the SQL buffer; SQL*Plus commands are not stored in the buffer.

Entering SQL*Plus Commands

SQL*Plus has its own commands to perform specific tasks on the database, as well as to format the query results. Unlike SQL statements, which are terminated with a semicolon or a blank line, SQL*Plus commands are entered on a single line. Pressing Enter executes the SQL*Plus command.

When you log in to the SQL*Plus session, you get the SQL prompt. By default, the prompt is `SQL>`. You can change this prompt using the `SET SQLPROMPT` SQL*Plus command. When you continue a SQL command to the next line, a line number appears at the beginning of the line. As shown here, when you type `SELECT USERNAME` in the first line (the SQL prompt line) and press Enter, line number 2 appears where you continue the SQL command `FROM DBA_USERS`.

```
SQL> SELECT USERNAME
   2 FROM DBA_USERS
```

SQL statements can span multiple lines. If you want to continue a SQL*Plus command onto the next line, you must end the current line with a hyphen (-), which indicates command continuation. When a command continuation character is entered, SQL*Plus will not show the line number next, but instead displays the greater than symbol (>). This is in contrast to SQL statements, which can be continued to the next line without a continuation operator. For example, the following SQL statement gives an error, because SQL*Plus treats the hyphen operator (-) as a continuation character instead of a minus operator:

```
SQL> SELECT 800 -
> 400 FROM dual;
SELECT 800  400 FROM dual
            *
ERROR at line 1:
ORA-00923: FROM keyword not found where expected
SQL>
```

You need to put the hyphen in the next line for the query to succeed:

```
SQL> SELECT 800
  2  - 400 FROM dual;

   800-400
----------
       400
SQL>
```

Getting Structural Information with the DESCRIBE Command

You can use the DESCRIBE command to obtain information about the database objects. Using DESCRIBE on a table or view shows the columns, its datatypes, and whether each column can be NULL. Using DESCRIBE on a stored program, such as procedure or function, shows the parameters that need to be passed in/out, their datatype, and whether there is a default value. You can abbreviate this command to the first four characters or more—DESC, DESCR, and DESCRIB are all valid.

If you're connected to the HR schema and need to see the tables and views in this schema, use the following query:

```
SQL> SELECT * FROM tab;

TNAME                          TABTYPE  CLUSTERID
------------------------------ ------- ----------
COUNTRIES                      TABLE
DEPARTMENTS                    TABLE
EMPLOYEES                      TABLE
EMP_DETAILS_VIEW               VIEW
JOBS                           TABLE
JOB_HISTORY                    TABLE
LOCATIONS                      TABLE
REGIONS                        TABLE

8 rows selected.
```

To see the columns or definition of the EMPLOYEES table, execute:

```
SQL> DESCRIBE employees
 Name                                      Null?    Type
 ----------------------------------------- -------- --------------------

 EMPLOYEE_ID                               NOT NULL NUMBER(6)
 FIRST_NAME                                         VARCHAR2(20)
```

```
LAST_NAME                                 NOT NULL VARCHAR2(25)
EMAIL                                     NOT NULL VARCHAR2(25)
PHONE_NUMBER                                       VARCHAR2(20)
HIRE_DATE                                 NOT NULL DATE
JOB_ID                                    NOT NULL VARCHAR2(10)
SALARY                                             NUMBER(8,2)
COMMISSION_PCT                                     NUMBER(2,2)
MANAGER_ID                                         NUMBER(6)
DEPARTMENT_ID                                      NUMBER(4)
```

If there are invisible columns in the table, they are not displayed by the `DESCRIBE` command unless you use `SET COLINVISIBLE ON`.

Invisible columns are newly introduced in Oracle Database 12*c*, where a column in the table can be hidden from the application. Invisible columns help to remove a column from the table quickly without actually dropping the column. Invisible columns are discussed in Chapter 7.

Editing the SQL Buffer

The most recent SQL statement executed or entered is stored in the SQL buffer of SQL*Plus. You can run the command in this buffer again by simply typing a slash or using the `RUN` command.

SQL*Plus provides a set of commands to edit the buffer. Suppose you want to add another column or add an `ORDER BY` condition to the statement in the buffer. You do not need to type the entire SQL statement again. Instead, just edit the existing statement in the buffer.

One way to edit the SQL*Plus buffer is to use the `EDIT` command to write the buffer to an operating-system file named `afiedt.buf` (this is the default filename, which can be changed) and then use a system editor to make changes.

You can use your favorite text editor by defining it in SQL*Plus. For example, to make Notepad your favorite editor, just issue the command `DEFINE _EDITOR = NOTEPAD`.

To view the editor defined, just execute `DEFINE _EDITOR` as shown here.

```
SQL> define _editor
DEFINE _EDITOR         = "Notepad" (CHAR)
```

Provide the entire path if the program is not available in the search path.

Another way to edit the buffer is to use the SQL*Plus editing commands. You can make changes, delete lines, add text, and list the buffer contents using the commands described in the following sections. Most editing commands operate on the current line. You can change

the current line simply by typing the line number. All commands can be abbreviated, except DEL (which is already abbreviated).

LIST

The LIST command lists the contents of the buffer. The asterisk indicates the current line. The abbreviated command for LIST is L.

```
SQL> L
  1  SELECT empno, ename
  2* FROM emp
SQL> LIST LAST
  2* FROM emp
SQL>
```

The command LIST *n* displays line *n*, and LIST * displays the current line. The command LIST *m n* displays lines from *m* through *n*. If you substitute * for *m* or *n*, it implies from or to the current line. The command LIST LAST displays the last line.

APPEND

The APPEND *text* command adds text to the end of a line. The abbreviated command is A.

```
SQL> A  WHERE empno <> 7926
  2* FROM emp WHERE empno <> 7926
SQL>
```

CHANGE

The CHANGE /*old*/*new* command changes an old entry to a new entry. The abbreviated command is C. If you omit *new*, *old* will be deleted.

```
SQL> C /<>/=
  2* FROM emp WHERE empno = 7926
SQL> C /7926
  2* FROM emp WHERE empno =
SQL>
```

The ellipses (...) can be used as wildcard characters. The following example changes everything in the line from "fro" to the new value.

```
SQL> l
  1* select name from v$instance
SQL> c/fro.../from v$database
  1* select name from v$database
SQL>
```

The next example shows the substitution of a string in the middle of the line using ellipses.

```
SQL> l
  1* select owner from dba_tables where table_name like 'HR%'
SQL> c/dba...table/dba_views where view
  1* select owner from dba_views where views where table_name like 'HR%'
SQL>
```

INPUT

The `INPUT` *`text`* command adds a line of text. Its abbreviation is `I`. If *`text`* is omitted, you can add as many lines as you want.

```
SQL> I
  3  7777 AND
  4  empno = 4354
  5
SQL> I ORDER BY 1
SQL> L
  1  SELECT empno, ename
  2  FROM emp WHERE empno =
  3  7777 AND
  4  empno = 4354
  5* ORDER BY 1
SQL>
```

DEL

The `DEL` command used alone or with `*` deletes the current line. The `DEL` *`m n`* command deletes lines from *`m`* through *`n`*. If you substitute `*` for *`m`* or *`n`*, it implies the current line. The command `DEL LAST` deletes the last line.

```
SQL> 3
  3* 7777 AND
SQL> DEL
SQL> L
  1  SELECT empno, ename
  2  FROM emp WHERE empno =
  3  empno = 4354
  4* ORDER BY 1
SQL> DEL 3 *
```

```
SQL> L
  1  SELECT empno, ename
  2* FROM emp WHERE empno =
SQL>
```

CLEAR BUFFER

The `CLEAR BUFFER` command (abbreviated `CL BUFF`) clears the buffer. This deletes all lines from the buffer.

```
SQL> L
  1  SELECT empno, ename
  2* FROM emp WHERE empno =
SQL> CL BUFF
buffer cleared
SQL> L
No lines in SQL buffer.
SQL>
```

Using Script Files

SQL*Plus provides commands to save the SQL buffer to a file, as well as to run SQL statements from a file. SQL statements saved in a file are called a *script file*.

You can work with script files as follows:

- To save the SQL buffer to an operating-system file, use the command `SAVE` *`filename`*. If you do not provide an extension, the saved file will have the extension `.sql`.
- By default, the `SAVE` command will not overwrite an existing file. If you want to overwrite an existing file, you need to use the keyword `REPLACE`.
- To add the buffer to the end of an existing file, use the `SAVE` *`filename`* `APPEND` command.
- You can edit the saved file using the `EDIT` *`filename`* command.
- You can bring the contents of a *script file* to the SQL buffer using the `GET` *`filename`* command.
- If you want to run a script file, use the command `START` *`filename`*. You can also run a script file using `@`*`filename`*.
- An `@@`*`filename`* used inside a script file looks for the filename in the directory where the parent *script file* is saved and executes it.

Exercise 1.1 will familiarize you with the script file commands, as well as the other topics covered so far.

EXERCISE 1.1

Practicing SQL*Plus File Commands

In this exercise, you will learn how to edit the SQL*Plus buffer using various buffer edit commands.

1. Enter the following SQL code; the third line is a blank line so that the SQL code is saved in the buffer:

```
SQL> SELECT employee_id, first_name, last_name
  2  FROM   employees
  3
SQL>
```

2. List the SQL buffer:

```
SQL> L
  1  SELECT employee_id, first_name, last_name
  2* FROM   employees
SQL>
```

3. Save the buffer to a file named `myfile`; the default extension will be `.sql`:

```
SQL> SAVE myfile
Created file MYFILE.sql
SQL>
```

4. Choose to edit the file:

```
SQL> EDIT myfile
SQL>
```

5. Add `WHERE EMPLOYEE_ID = 106` as the third line to the SQL statement.
6. List the buffer:

```
SQL> LIST
  1  SELECT employee_id, first_name, last_name
  2* FROM   employees
SQL>
```

   The buffer listed is still the old buffer. The edited changes are not reflected because you edited the file `MYFILE`, which is not yet loaded to the buffer.

EXERCISE 1.1 *(continued)*

7. Bring the file contents to the buffer:

```
SQL> GET myfile
  1  SELECT employee_id, first_name, last_name
  2  FROM   employees
  3* WHERE employee_id = 106
SQL>
```

8. List the buffer to verify its contents:

```
SQL> LI
  1  SELECT employee_id, first_name, last_name
  2  FROM   employees
  3* WHERE employee_id = 106
SQL>
```

9. Change the employee number from 106 to 110:

```
SQL> C/106/110
  3* WHERE employee_id = 110
SQL>
```

10. Save the buffer again to the same file:

```
SQL> SAVE myfile
SP2-0540: File "MYFILE.sql" already exists.
Use "SAVE filename[.ext] REPLACE".
SQL>
```

An error is returned, because `SAVE` will not overwrite the file by default.

11. Save the file using the `REPLACE` keyword:

```
SQL> SAVE myfile REPLACE
Wrote file MYFILE.sql
SQL>
```

12. Execute the file:

```
SQL> START myfile
```

EXERCISE 1.1 *(continued)*

```
EMPLOYEE_ID FIRST_NAME           LAST_NAME
----------- -------------------- ---------
        110 John                 Chen
SQL>
```

13. Change the employee number from 110 to 106, and append this SQL code to the file; then execute it using `@`:

```
SQL> C/110/106
  3* WHERE employee_id = 106
SQL> SAVE myfile APPEND
Appended file to MYFILE.sql
SQL> @MYFILE
EMPLOYEE_ID FIRST_NAME           LAST_NAME
----------- -------------------- ---------
        110 John                 Chen


EMPLOYEE_ID FIRST_NAME           LAST_NAME
----------- -------------------- ---------
        106 Valli                Pataballa
SQL>
```

Saving Query Results to a File

You can use the `SPOOL` *`filename`* command to save the query results to a file. By default, the `SPOOL` command creates a `.lst` file extension. `SPOOL` overwrites an existing file by default. If you include the `APPEND` option—as in `SPOOL` *`filename`* `APPEND`—the results are added to an existing file. A new file will be created if the file does not exist already.

`SPOOL OFF` stops writing the output to the file. `SPOOL OUT` stops the writing of output and sends the output file to the printer. `SPOOL` with no clauses lists the current spooling status.

Adding Comments to a Script File

Comments in the script file can improve readability and make the code more understandable. You can enter comments in SQL*Plus using the `REMARKS` (abbreviated `REM`) command. Lines in the script file beginning with the keyword `REM` are comments and are not executed. You can also enter a comment between `/*` and `*/`. Comments can also be entered following `--` (double hyphen); all characters following `--` in the line are treated as comments by Oracle.

While a script file with comments is being executed, the remarks entered using the `REMARKS` command are not displayed on the screen, but the comments within `/*` and `*/` are displayed on the screen with the prefix `DOC>` when there is more than one line between `/*` and `*/`. You can turn this off by using `SET DOCUMENT OFF`.

Now that you understand the concepts of RDBMS and how Oracle Database 12*c* helps organizations achieve the cloud architecture, let's move on to the core of the Oracle Database 12*c* SQL Fundamentals OCA exam in the coming chapters. Before moving on to Chapter 2, "Introducing SQL," please make sure you have an Oracle Database 12*c* to practice on and try out the examples.

You may perform a quick default install of the database after downloading the software from OTN (`www.technet.oracle.com`).

Real World Scenario

Install Oracle Database 12*c* for SQL Practice

To be able to practice the examples provided in this book and to familiarize yourself with Oracle Database 12*c* SQL, an Oracle Database 12*c* database must be available to you. If you do not have such a database, you can follow these instructions to install software and create databases on a Windows machine.

Download and Install Software

You may download Oracle Database 12*c* software from Oracle Technology Network (OTN) or from Oracle Cloud Delivery service (`edelivery.oracle.com`). After downloading the software, you can invoke the `setup.exe` to install software. For detailed instructions on downloading and installing Oracle software, refer to `www.bijoos.com/certify/db12csw.pdf`. You can also refer to Chapter 9 to install database software.

Create Oracle Database

Databases are created using the Database Configuration Assistant tool. You can choose the Create Database With Default Configuration option to create a database quickly. For detailed instructions on creating a database, refer to `www.bijoos.com/certify/db12c_ndb.pdf`. You can also refer to Chapter 9 to create a database.

Create Sample Schema

The sample schema provided by Oracle includes HR, OE, PM, SH, and IX. For the majority of the SQL used in the book, the HR schema is used. If you did not install the sample schema during database creation, you can do so using the following procedure.

When you install Oracle software, you can choose the Create Database With Default Configuration option, but this will not include the sample schemas. The account `SYS` is the Oracle dictionary owner, and `SYSTEM` is a database administrator (DBA) account. Initially, the sample schema user accounts are locked. You need to log in to the database using SQL*Plus as the `SYSTEM` user and then unlock the account using the `ALTER USER` statement. To unlock the `HR` schema, use `ALTER USER hr IDENTIFIED BY hrpassword ACCOUNT UNLOCK;`. Now you can log in to the database using the `hr` user with the password `hrpassword`. Remember, the password is case sensitive by default.

To install the sample schemas in an existing Oracle Database 12*c*, follow the instructions in the Oracle document "Oracle Database Sample Schemas 12*c* Release 1 (12.1) Part E15979-04" at `http://docs.oracle.com/`. Chapter 2 of this document provides instructions on how to install the sample schemas using Database Configuration Assistant (DBCA) as well as on running scripts. The same chapter also gives you steps to reinitialize the sample schema data.

The manual installation of HR and OE sample data on Linux-based Oracle Database 12*c* databases can be quickly summarized as:

Change the directory to `$ORACLE_HOME/demo/schema/human_resources`.

Connect to database using SQL*Plus as SYSDBA (`sqlplus sys@mydb as sysdba`).

Run the schema and objects creation script (`@hr_main.sql`).

Change the directory to `$ORACLE_HOME/demo/schema/order_entry`.

Connect to the database using SQL*Plus as SYSDBA (`sqlplus sys@mydb as sysdba`).

Run the schema and objects creation script (`@oe_main.sql`).

Summary

This chapter reviewed the concepts of relational database systems and Object RDBMS. You also learned how Oracle implements the RDBMS and relational theory into the Oracle database. The entity-relationship diagram is a modeling tool used in the beginning stages or application development.

You also learned about the high-level architecture and various implementations of Oracle, such as single database, RAC database, and container database.

Oracle Database 12*c* is cloud enabled. The multitenant architecture of the database helps to consolidate multiple Oracle databases (pluggable databases) into a single container database.

Various tools are available for the DBA to connect to the Oracle database and administer it. SQL*Plus is Oracle's SQL command-line interface tool. SQL Developer is a graphical tool, with ease of navigation and predefined tasks. You also saw an overview of SQL*Plus in this chapter, including how to connect to the database using SQL*Plus and basic editing commands.

SQL*Plus supports all SQL statements and has its own formatting and enhancement commands. Using this tool, you can produce interactive SQL statements and formatted reports. SQL*Plus is the command-line interface to the database widely used by DBAs. SQL*Plus has its own buffer where SQL statements are buffered. You can edit the buffer using SQL*Plus editing commands. The `DESCRIBE` command is used to get information on a table, view, function, or procedure. Multiple SQL and SQL*Plus commands can be stored in a file and can be executed as a unit. Such files are called script files.

Exam Essentials

Know RDBMS Concepts. Review the RDBMS concepts. Understand entities and relationships.

Understand what structures make Object RDBMS. Learn how Oracle implements the object relational database management system.

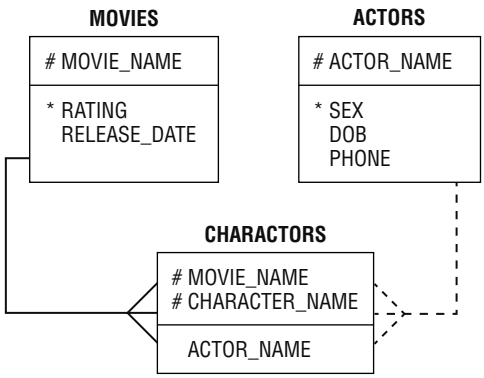
Know the tools. Have an understanding of what tools are available for database management in Oracle and their purposes.

Learn the various architectures Oracle Database 12*c* can implement. Oracle database can be installed as a single instance single database, multiple instance RAC database, or multitenant container database.

Identify Oracle Database 12*c* cloud features. Know the features of Oracle Database 12*c* that make cloud implementation easier.

Review Questions

1. Look at the diagram. What kind of relationship exists between MOVIES and CHARACTERS?



   A. Each movie may have one or more characters.

   B. Each movie must have one or more characters.

   C. Many movies may have many characters.

   D. One movie can have only one character.

2. When the physical model is being designed from the logical model, which element may be attributed as a table from the ER diagram?

   A. Relationship

   B. Attribute

   C. Unique identifier

   D. Entity

3. Which statement about the object type is true?

   A. They are structures that consist of built-in or user-defined data types.

   B. They are structures that consist of only built-in data types.

   C. They are structures that consist of only user-defined data types.

   D. Only one column in a table can be object type.

4. Which of the following is not a benefit of Oracle Database 12*c*?

   A. Manage multiple databases as one

   B. Fast provisioning of cloned databases

   C. Plug and unplug databases

   D. Patch each pluggable database separately

5. Which one of the following Oracle SQL*Plus command lines is not valid?
   A. `sqlplus <username>`
   B. `sqlplus @<connect_string>`
   C. `sqlplus <username>@<connect_string>`
   D. `sqlplus`
6. Which database tools are parts of Oracle Database 12*c*? Choose two.
   A. Oracle Enterprise Manager Cloud Control 12*c*
   B. Oracle Enterprise Manager Database Express 12*c*
   C. SQL Developer
   D. TOAD (Tool for Oracle Application Developers)
7. In the physical implementation of RDBMS, which database object is used to represent unique identifiers?
   A. Any constraint
   B. Index
   C. Primary key
   D. Foreign key
8. SQL Developer is a tool primarily for whom?
   A. Database administrators
   B. End users
   C. Application developers
   D. All of the above
9. Which architecture in the Oracle Database 12*c* implementation guards against unplanned machine downtime?
   A. Multitenancy Container Database
   B. Real Application Clusters
   C. Consolidate multiple databases and instances to one server
   D. None of the above
10. Which connection method to the Oracle database is known as the easy connect?
    A. `<username>@<connect_string>`
    B. `<username>@<host>:<port>/<service_name>`
    C. Both A and B
    D. Neither A or B