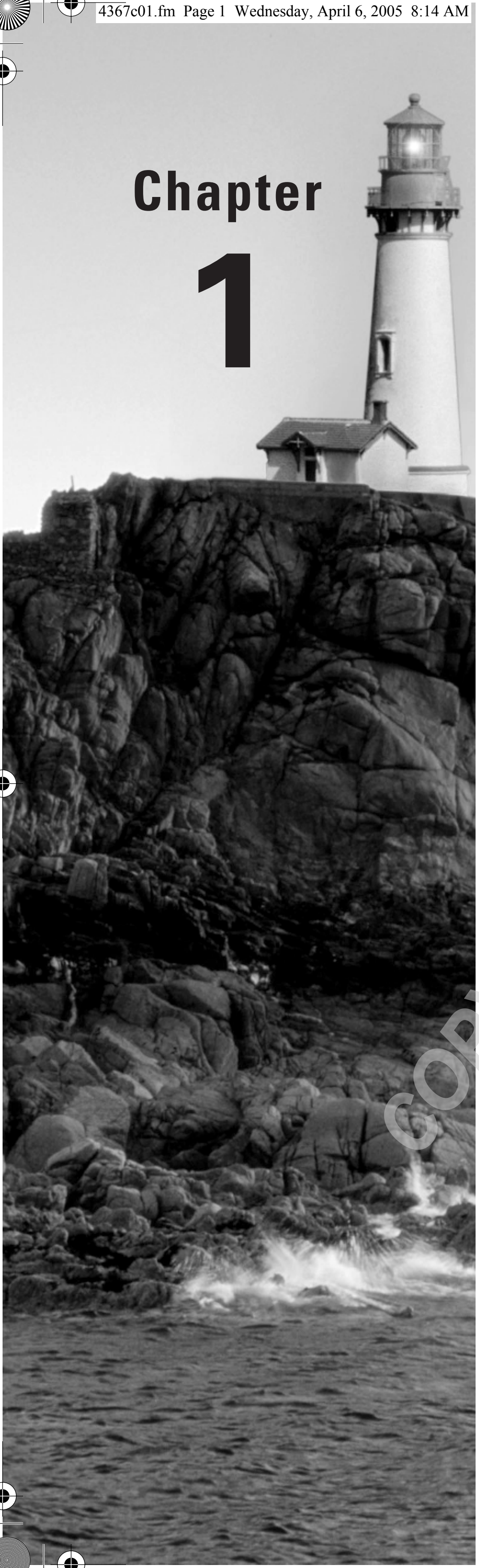


Oracle Database 10*g* Components and Architecture

ORACLE DATABASE 10*G*: ADMINISTRATION I EXAM OBJECTIVES COVERED IN THIS CHAPTER:

✓ **Installing Oracle Database 10*g* Software**
- Identify system requirements.
- Use Optimal Flexible Architecture.
- Install software with Oracle Universal Installer.
- Identify and configure commonly used environment variables.

✓ **Creating an Oracle Database**
- Explain the Oracle database architecture.
- Explain the instance architecture.

✓ **Database Interfaces**
- Use SQL*Plus and iSQL*Plus to access the Oracle Database 10*g*.
- Use SQL*Plus and iSQL*Plus to describe the logical structure of tables.
- Use SQL to query, manipulate, and define data using SELECT, UPDATE/INSERT/DELETE and CREATE/ALTER/DROP statements.
- Identify common database interfaces.
- Describe a database transaction.

NOTE: 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 (`http://www.oracle.com/education/certification/`) for the most current exam objectives listing.

With the release of Oracle Database 10*g* (Oracle 10*g*), Oracle Corporation has delivered a powerful and feature-rich database that can meet the performance, availability, recoverability, and security requirements of any mission-critical application. As the Oracle DBA, you are responsible for managing and maintaining the Oracle 10*g* database from initial installation, creation, and configuration to final deployment. 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.

This chapter introduces you to important concepts associated with the relational nature of Oracle 10*g* and its architecture. This chapter will also help you learn the specifics of how to install, configure, and manage an Oracle database. You need a solid understanding of these concepts before moving on to subsequent chapters.

The Oracle Product Family

Oracle Corporation is generally thought of as a database company—and for good reason. Oracle has been a leader in the development of reliable, scalable, and recoverable database technologies for more than 25 years. With the release of Oracle 10*g*, Oracle has further enhanced its reputation as an industry leader by producing a feature-rich, yet easy-to-manage database that can handle data from the busiest transactional systems to the largest data warehouses. However, Oracle Corporation also produces many other products that support a variety of data-related business activities. Currently, Oracle's product family consists of the following products and services:

- Oracle Database 10*g*
- Oracle Application Server 10*g*
- Oracle Developer Suite
- Oracle Applications 11*i*
- Oracle Collaboration Suite
- Oracle Services

Each of these products or services is described in detail in the following sections.

NOTE The following information regarding Oracle's products is included to provide a framework for understanding where Oracle 10*g* fits within the larger Oracle product line. This information is not part of the exam objectives.

Oracle 10*g*

Oracle 10*g* was released as version 10.1.0.2 in spring 2004. This release of Oracle's flagship database product introduces many new features, but the three primary thrusts are ease of management, enhanced scalability, and improved performance management.

The ease of management features include the automatic management of disk storage allocated to the database, proactive monitoring and self-tuning of the database's memory structures, preconfigured database alerts, and enhanced, web-based tools for monitoring and managing the entire Oracle architecture.

Scalability and performance improvements are largely based on Oracle's grid computing model. *Grid computing* is intended to allow businesses to move away from the idea of many individual servers, each of which is dedicated to a small number of applications. When configured in this manner, applications often either do not fully utilize the server's available hardware resources such as memory, CPU, and disk or fall short of these resources during peak usage. By comparison, databases running under Oracle's grid computing model can be spread across as few or as many servers as needed so as to make the most efficient use of each of the available hardware resources at all times. At the same time, Oracle 10*g*'s automated performance monitoring and tuning mechanisms dynamically adjust the database's allocation of these resources to improve performance.

There are five editions of Oracle 10*g*:

- Enterprise
- Standard
- Standard Edition One
- Personal
- Lite

Table 1.1 compares these versions.

TABLE 1.1 Comparison of Oracle 10*g* Editions

Edition	Description
Enterprise Edition	Includes all available Oracle 10*g* features either bundled or as extra-cost options.
Standard Edition	Includes full clustering features and all Oracle 10*g* ease-of-management features for servers running as many as four processors.

TABLE 1.1 Comparison of Oracle 10*g* Editions *(continued)*

Edition	Description
Standard Edition One	Includes all Oracle 10*g* ease-of-management features for servers running as many as two processors.
Personal	Includes all available Oracle 10*g* features either bundled or as extra-cost options, but for an individual user database.
Lite	Includes all Oracle 10*g* features needed to build and deploy mobile database applications.

NOTE Most of the examples in this book are based on the Enterprise Edition.

Oracle Application Server 10*g*

Oracle's Application Server 10*g* is used to deploy web-based applications that, like the database, must be highly reliable and scalable to thousands of users. Like the database, Oracle Application Server 10*g* is also available in a number of versions, but all versions include full Java functionality and Oracle's HTTP server, with additional components such as portals, forms and reports servers, single sign-on capabilities, and wireless connectivity also available.

Oracle Developer Suite

Oracle's Developer Suite consists of several products that can be used to design, develop, and distribute web-based applications. These tools include the following:

- Oracle Designer for gathering business requirements and designing applications
- Oracle JDeveloper for creating Java-based applications
- Oracle Forms and Reports Developer for creating and deploying custom forms and reports
- Oracle Discoverer for developing and distributing ad hoc reporting capabilities against application data
- Oracle Warehouse Builder for designing and deploying data marts and warehouses.

Each of these tools is designed to integrate seamlessly with the Oracle database and application server products to provide a robust application development environment.

Oracle Applications 11*i*

Oracle's database, application server, and developer products make up the core infrastructure of the E-Business Suite of products collectively called Oracle Applications 11*i*. Oracle Applications 11*i* is composed of a number of modules that are used to manage the financial, personnel, manufacturing, order management, sales, service, and asset data of both businesses and public sector organizations.

Oracle Collaboration Suite

Oracle's Collaboration Suite offers a comprehensive system that integrates all of a business's communication technologies, from e-mail, voice mail, and faxes to wireless connectivity and web conferencing. Like Oracle Applications 11*i*, the Collaboration Suite also uses Oracle's database and application server technologies as the core technology infrastructure. This provides a scalable and reliable platform for true enterprise-wide collaboration.

Oracle Services

In addition to software development, Oracle also offers a variety of technical support and consulting services. Technical support is delivered primarily through Oracle's MetaLink website and is available to all customers with current maintenance agreements. In addition to this support, Oracle Services also offers consulting services to help customers select, install, and configure the Oracle technologies that best meet their needs.

NOTE

You can access Oracle's MetaLink support site at `http://metalink.oracle.com`. This site provides a wealth of patches, documentation, notes, white papers, and user forums.

NOTE

A valid Custom Support Identifier(CSI) is required to create a MetaLink account. A unique CSI number is usually issued for each Oracle product that is purchased.

Another service offered by Oracle is education. Oracle develops and delivers instructor-led and web-based training courses for all their products. These courses are taught at Oracle University sites and Oracle Approved Education Center locations throughout the world. Oracle Education is also responsible for coordinating all of Oracle's certification programs, including the Oracle Database 10*g* Oracle Certified Associate (OCA) and Oracle Certified Professional (OCP) certifications for which this book helps you prepare.

Relational Database Concepts

At the heart of all the Oracle products discussed in the previous section is the concept of using a database to store, manipulate, retrieve, and secure important business data. The manner in which these three tasks are performed has varied throughout the history of computing. 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.

Oracle 10*g* does not use any of these techniques for storing or accessing data. Instead, all releases of Oracle's database products have used a relational model to store application data in the database. This relational model on which Oracle is built is based on the ground-breaking work of Dr. Edgar Codd, which was first published in 1970 in his paper "A Relational Model of Data for Large Shared Data Banks."

NOTE Oracle Corporation (then known as Relational Software, Inc.) released the first commercially available relational database in 1979.

IBM Corporation was an early adopter of Dr. Codd's model and also helped to develop the computer language that is used to access all relational databases today—*Structured Query Language (SQL)*. Using English-like commands, SQL users can easily interact with relational databases without having to write complex computer programs or needing to know where or how the data is physically stored on disk. Samples of SQL statements are used in examples throughout this book. In general, SQL commands are used to do the following:

- Display data stored in database tables using the `SELECT` command
- Add rows to tables using the `INSERT` command
- Remove rows from tables using the `DELETE` command
- Modify rows in tables using the `UPDATE` command
- Create, modify, or drop tables using the `CREATE`, `ALTER`, and `DROP` commands
- Grant or revoke user access to tables using the `GRANT` and `REVOKE` commands
- Control transactions using the `COMMIT` and `ROLLBACK` commands

Even though each of the previous commands is an SQL command, each type of SQL statement can be classified into one of four categories:

- *Queries* using the `SELECT` command.
- Statements using the `CREATE`, `ALTER`, or `DROP` command are classified as *Data Definition Language (DDL)* commands.
- Statements using the `GRANT` or `REVOKE` commands are classified as *Data Control Language (DCL)* commands.
- Statements using the `INSERT`, `UPDATE`, and `DELETE` commands are classified as *Data Manipulation Language (DML)* commands.

DML commands are used in transactions. A *transaction* begins with the first DML command that a user issues and ends when the user either makes their changes permanent by issuing a `commit` command or undoes their changes using the `rollback` command.

NOTE Issuing a DDL or a DCL command also ends any prior transactions by causing an implicit `commit` command to occur. Abnormal terminations of a database connection to a network or a power failure can cause implicit rollbacks to occur.

Most SQL statements, whether they are queries, DMLs, DDLs, or DCLs, are directed at data stored in one or more Oracle tables. The next section examines important Oracle table concepts in detail.

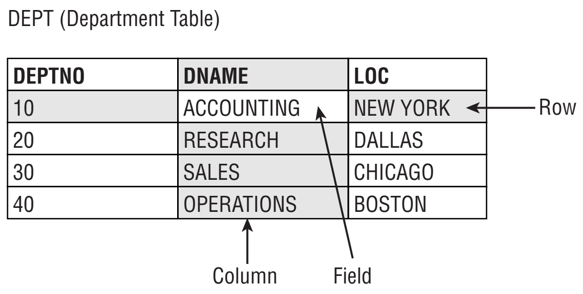
Rows, Columns, Tables, and Databases

At the heart of the relational model is the concept of a table. A table is composed of columns and rows. The intersection of a column and a row is called a field. The collection of tables that store business data are stored within the Oracle 10*g* database. Figure 1.1 shows an example of a table, a column, a row, and a field for a table called `DEPT` that stores department data.

The `DEPT` table in Figure 1.1 is composed of three columns (`DEPTNO`, `DNAME`, and `LOC`) and contains four rows. Each row contains all the relevant data for a single department. The field at the intersection of the `DNAME` column and the first row contains the value "Accounting". When a table is created, each column is assigned a name and a datatype. Many datatypes are available in Oracle 10*g*, but most simply designate whether a column is intended to store characters, numbers, or dates. You can use the following DDL statement to create the `DEPT` table shown in Figure 1.1.

```
SQL> create table DEPT
  2  (DEPTNO  number(2),
  3   DNAME   varchar2(14),
  4   LOC     varchar2(13));

Table created.
```

FIGURE 1.1 An example of a table composed of columns, rows, and fields



DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

The DDL command creates a column called DEPTNO to store department numbers of as many as 2 digits, a column called DNAME to store department name data of as many as 14 characters, and a column called LOC to store department location data of as many as 13 characters. By specifying column datatypes in this manner, some basic data controls are automatically in place within the database. These controls prevent a user from storing incorrect data in a table. For example, attempting to insert a record that stores a word in a column that is set up to hold numeric values causes a SQL error. The following example shows an example of an INSERT statement that succeeds because all the data being inserted is of the correct datatype, and it shows another statement that fails because a character datatype was inserted into the numeric DEPTNO column:

```
SQL> insert into DEPT (DEPTNO, DNAME, LOC)
  2   values (50,'MANUFACTURING','MADISON');

1 row created.

SQL> insert into DEPT (DEPTNO, DNAME, LOC)
  2   values ('SIX','SHIPPING','MILWAUKEE');
values ('SIX','SHIPPING','MILWAUKEE')
        *
ERROR at line 2:
ORA-01722: invalid number
```

NOTE These are simplified examples. Oracle 10*g* can accommodate tables that have as many as 1,000 columns and billions of rows.

NOTE See Chapter 3, “Database Storage and Schema Objects,” for more information about creating tables and other database objects.

In addition to tables such as DEPT that store important business data, Oracle databases also contain system tables that store data about the database itself. Examples of the type of information in these system tables include the names of all the tables in the database, the column names and datatypes of those tables, the number of rows those tables contain, and security information about which users are allowed to access those tables. This “data about the database” is referred to as *metadata*. As a DBA, you will frequently use this metadata when performing your tasks.

The metadata tables, however, have rather cryptic names such as OBJ$, FILE$, X$KSMSP, and X$KWQSI with unusual column names such as DATAOBJ#, CRSCNWRP, KSMCHCOM, and KWQSINCO. To make it easier to use SQL to examine the contents of metadata tables, Oracle builds views on the tables. A *view* is similar to a table in that it is made up of columns and rows. However, a view is only a logical structure that contains no data of its own. Instead, a view is like a window that can be used to look at the contents of another table or tables. Views greatly simplify access to the metadata because the names of the views and the columns in them are much more intuitive than the metadata tables on which they are based. An Oracle 10*g* database contains two types of metadata views:

- Data dictionary views
- Dynamic performance views

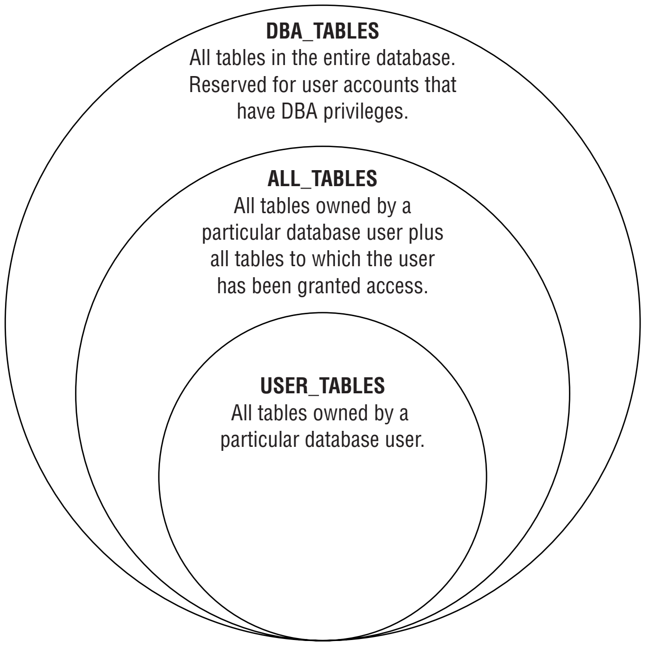
Examples of both data dictionary and dynamic performance views are described in the next section.

Data Dictionary Views

Depending on which features are installed and configured, an Oracle 10*g* database can contain more than 1,300 data dictionary views. Data dictionary views have names that begin with DBA_, ALL_, and USER_.

The difference between the DBA_, ALL_, and USER_ views can be illustrated using the DBA_TABLES data dictionary view as an example. The DBA_TABLES view shows information on *all* the tables in the database. The corresponding ALL_TABLES view, despite its name, shows only the tables that a particular database user owns or has access to. For example, if you were logged in to the database as a user named SCOTT, the ALL_TABLES view would show all the tables owned by the user SCOTT and the tables to which SCOTT has been granted access by other users. The USER_TABLES view shows only those objects owned by a user. If the user SCOTT were to examine the USER_TABLES view, only those tables he owns would be displayed. Figure 1.2 shows a graphical representation of the relationship between the DBA_, ALL_, and USER_views.

FIGURE 1.2 A comparison of data dictionary views


DBA_TABLES
All tables in the entire database. Reserved for user accounts that have DBA privileges.
ALL_TABLES
All tables owned by a particular database user plus all tables to which the user has been granted access.
USER_TABLES
All tables owned by a particular database user.

Because the DBA_ views provide the broadest metadata information, they are generally the data dictionary views used by DBAs. Table 1.2 provides more examples of DBA_ data dictionary views.

TABLE 1.2 Examples of Data Dictionary Views

Dictionary View	Description
`DBA_TABLES`	Shows the names and physical storage information about all the tables in the database.
`DBA_USERS`	Shows information about all the users in the database.
`DBA_VIEWS`	Shows information about all the views in the database.
`DBA_TAB_COLUMNS`	Shows all the names and datatypes of the table columns in the database.

NOTE A complete list of the Oracle 10*g* data dictionary views can be found in Chapters 2 and 3 of the *Oracle Database Reference 10*g *Release 1 (10.1) Part Number B10755-01* available at `http://tahiti.oracle.com`.

Dynamic Performance Views

Depending on which features are installed and configured, an Oracle 10*g* database can contain approximately 350 dynamic performance views. Most of these views have names that begin with V$. Table 1.3 describes a few of these dynamic performance views.

TABLE 1.3 Examples of Dynamic Performance Views

Dynamic Performance View	Description
`V$DATABASE`	Contains information about the database itself, such as the database name and when the database was created.
`V$VERSION`	Shows which software version the database is using.
`V$OPTION`	Displays which optional components are installed in the database.
`V$SQL`	Displays information about the SQL statements that database users have been issuing.

NOTE A complete list of the Oracle 10*g* data dictionary views can be found in Chapter 4 of the *Oracle Database Reference 10*g *Release 1 (10.1) Part Number B10755-01* available at `http://tahiti.oracle.com`.

Although the contents of the `DBA_` and `V$` metadata views are similar, there are some important differences between the two types. Table 1.4 compares these two types.

TABLE 1.4 A Comparison of Data Dictionary and Dynamic Performance Views

Dictionary Views	Dynamic Performance Views
The `DBA_` views usually have plural names (for example, `DBA_DATA_FILES`).	The names of the `V$` views are generally singular (for example, `V$DATAFILE`).
The `DBA_` views are available only when the database is open and running.	Some `V$` views are available even when the database is not fully open and running.
The data contained in the `DBA_` views is generally uppercase.	The data contained in the `V$` views is usually lowercase.
The data contained in the `DBA_` views is static and is not cleared when the database is shut down.	The `V$` views contain dynamic statistical data that is lost each time the database is shut down.

NOTE As an alternative to querying data dictionary and dynamic performance views directly, you can use the web-based Oracle Enterprise Manager Database Control tools to graphically display metadata information.

Data dictionary views are useful for examining the relationships between tables and the rules defined for storing data in tables. These restrictions and relationships are examined in the next section.

Relationships and Constraints

Real-world Oracle databases are made up of hundreds or thousands of tables. To use these tables to more easily store and retrieve data, you can define rules about how the tables are related and how data should be stored in each table. These rules are referred to as constraints. A *constraint* allows the database designer to enforce business rules about the data stored in the database's tables and the relationships between tables. Table 1.5 describes the five types of constraints in an Oracle database.

TABLE 1.5 Types of Table Constraints

Constraint Type	Description
`Not Null`	A value must be supplied for this column, but values do not have to be unique.
Unique Key	Every value in this column must be unique, but null values are allowed.
Primary Key	Every value in the column must be unique and cannot be null.
Foreign Key	Every value in the column must match a value in another column in this table or some other table; otherwise, the value is null.
Check	The value entered in the table must match one of the specified values for this column.

NOTE A null value is the absence of any value; it is not the same as a space or a zero.

NOTE Constraint information is stored in the `DBA_CONSTRAINTS` and `DBA_CONS_COLUMNS` data dictionary views.

For example, suppose your database contains a table called `EMP` that holds employee information. Table 1.6 shows the structure of the `EMP` table.

TABLE 1.6 The Structure of an *EMP* Table

Column Name	Column Description	Column Datatype
`EMPNO`	Employee ID number	Number
`ENAME`	Employee name	Character
`JOB`	Job title	Character
`MGR`	Manager's employee ID	Number
`HIREDATE`	Date employee was hired	Date
`SAL`	Employee's monthly salary	Number

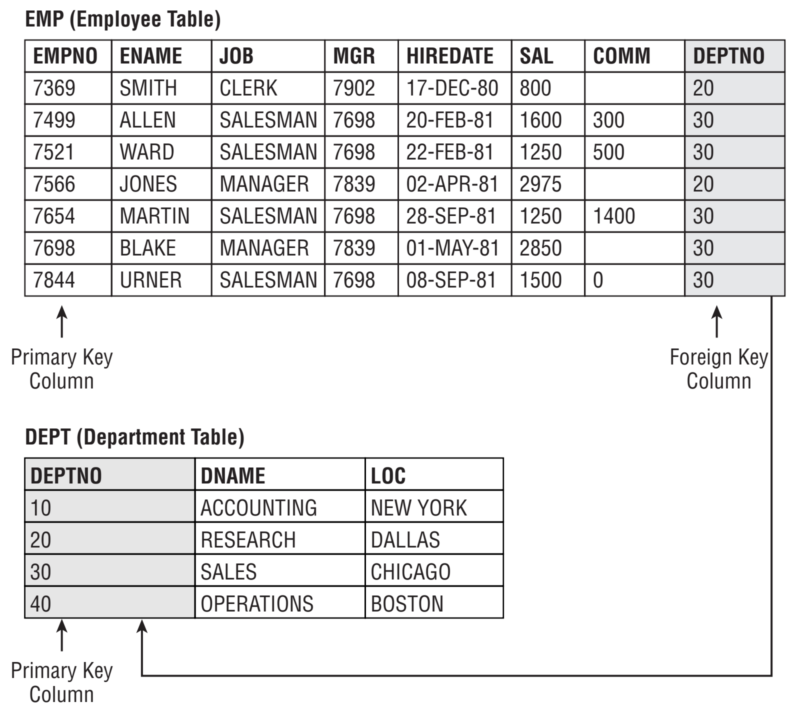
TABLE 1.6 The Structure of an *EMP* Table *(continued)*

Column Name	Column Description	Column Datatype
COMM	Employee's commission amount	Number
DEPTNO	Employee's department number	Number

If the business has a rule that every employee must have an employee ID and that no two employee IDs can be the same, placing a primary key constraint on the EMPNO column of the EMP table enforces this rule. Any records inserted without an employee number, or with the same employee number as an existing employee, are rejected. Therefore, the EMPNO column is referred to as the *primary key* of the EMP table because the EMPNO value uniquely identifies each record in the EMP table.

A business rule might require that each employee be assigned to a valid department. To enforce this rule, you can define a foreign key constraint between the EMP and DEPT tables so that the DEPTNO value entered for every employee in the EMP table must have a matching DEPTNO in the DEPT table. This relationship is shown graphically in Figure 1.3.

FIGURE 1.3 The relationship between the *EMP* and *DEPT* tables



EMP (Employee Table)

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7844	URNER	SALESMAN	7698	08-SEP-81	1500	0	30

DEPT (Department Table)

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

In this example, the DEPTNO column of the EMP table is referred to as a *foreign key* because it has a relationship to the DEPTNO column in another (that is, foreign) table called DEPT. Designing database tables in this manner, so that the values in one table have a relationship to the values in another table, is referred to as *referential integrity (RI)*. Referential integrity is generally enforced through the use of primary key and foreign key table constraints.

NOTE

In addition to defining relationships between tables, you can also use foreign keys to define relationships between two columns within the same table. These types of constraints are referred to as *self-referencing foreign keys*.

Sample DDL commands to create the DEPT and EMP tables with the primary and foreign key constraints that we've described are shown here:

```
SQL> alter table DEPT
  2  add constraint DEPT_PK
  3  primary key (DEPTNO);
Table altered.

SQL> create table EMP
  2  (empno    number(4) constraint EMP_PK primary key,
  3  ename    varchar2(10),
  4  job      varchar2(9),
  5  mgr      number(4),
  6  hiredate date,
  7  sal      number(7,2),
  8  comm     number(7,2),
  9  deptno   number(2) constraint EMP_PK_DEPTNO references DEPT(deptno)
 10  );

Table created.
```

Notice that because the DEPT table did not have a primary key defined when we originally created it, the ALTER command is used to create one. Once the relationship between the two tables is defined, the database enforces the relationship for every DML statement performed on those tables. The following example shows an INSERT into the EMP table that fails because the DEPT table has no corresponding department record; it also shows how the same INSERT succeeds after the proper foreign key record is present in the DEPT table:

```
SQL> insert into EMP (empno, ename, deptno)
  2  values (84,'JOHNSON',99);
insert into EMP (empno, ename, deptno)
*
ERROR at line 1:
```

```
ORA-02291: integrity constraint (SCOTT.EMP_PK_DEPTNO) violated - parent key
     not found

SQL> insert into DEPT (deptno, dname, loc)
  2  values (99,'RESEARCH','FREEPORT');

1 row created.

SQL> insert into EMP (empno, ename, deptno)
  2  values (84,'JOHNSON',99);

1 row created.
```

Referential integrity not only enforces the relationship rules while rows are added to a table, but also enforces those rules when rows are being deleted or updated as well. For example, if a user attempts to delete a department from the DEPT table, that department must not have any employees assigned to it; if it does, the primary key/foreign key relationship will not allow the delete. If the employees in that department are deleted first, the DELETE statement on the DEPT table succeeds. The following example demonstrates this behavior when a DELETE statement is issued on tables with referential constraints:

```
SQL> delete from DEPT
  2  where deptno = 99;
delete from DEPT
*
ERROR at line 1:
ORA-02292: integrity constraint (SCOTT.EMP_PK_DEPTNO) violated - child
     record found


SQL> delete from EMP
  2  where deptno = 99;

1 row deleted.

SQL> delete from DEPT
  2  where deptno = 99;

1 row deleted.
```

NOTE

The ORA-02292 error can be avoided if a foreign key constraint is defined with the ON DELETE CASCADE option. Defining a foreign key in this manner causes Oracle 10*g* to automatically delete child records when a parent record is deleted.

Constraints have a similar impact on UPDATE statements. If a department's number is updated, the database determines if there are employees in that department before allowing the update. If there are employees in that department, the UPDATE fails, because changing the department number will "orphan" these employees, leaving them without a valid department—which violates the business rule that the constraint was designed to enforce. The following example shows what happens when an UPDATE violates the RI rules in the database:

```
SQL> update DEPT
  2  set deptno = 1
  3  where deptno = 10;
update DEPT
*
ERROR at line 1:
ORA-02292: integrity constraint (SCOTT.EMP_PK_DEPTNO) violated - child
     record found
```

Constraints also prevent a user from removing a table that has a defined relationship to another table. The following example shows how RI impacts an attempt to use the SQL DROP command on the DEPT table:

```
SQL> drop table DEPT;
drop table DEPT
           *
ERROR at line 1:
ORA-02449: unique/primary keys in table referenced by foreign keys
```

When two tables share a common column, such as when referential integrity constraints are defined on the columns between two tables, you can join those tables in a query and return rows from both tables simultaneously. The relationship between the two tables is defined in the WHERE clause of the query shown here.

```
SQL> select dname, ename
  2  from DEPT, EMP
  3  where DEPT.deptno = EMP.deptno;

DNAME          ENAME
-------------- ---------------
ACCOUNTING     CLARK
ACCOUNTING     KING
ACCOUNTING     MILLER
RESEARCH       SMITH
RESEARCH       ADAMS
RESEARCH       FORD
RESEARCH       SCOTT
RESEARCH       JONES
SALES          ALLEN
SALES          BLAKE
SALES          MARTIN
SALES          JAMES
SALES          TURNER
SALES          WARD

14 rows selected.
```

This query joins the two tables on the common `DEPTNO` column. Because the `DEPTNO` column has the same name in both tables, each table's name is included in `JOIN` condition of the `WHERE` clause to explicitly tell Oracle how to perform the `JOIN`.

NOTE

In addition to the traditional Oracle `JOIN` syntax shown in the previous example, Oracle 10*g* is also fully compliant with the ANSI SQL: 1999 syntax that uses the `JOIN`, `CROSS JOIN`, or `NATURAL JOIN` keywords when joining tables.

NOTE

If you include two or more tables in the `FROM` clause, but forget to join the tables in the `WHERE` clause, the query produces a *Cartesian product*. Cartesian products simply join every row in the first table to every row in the second table without regard for the defined relationship between tables—usually producing a meaningless, I/O-intensive result.

Other Segment Types

The previous section described a variety of SQL commands that can be used against tables in the database. However, tables are just one type of segment in an Oracle 10*g* database. A *segment* is defined as any entity that consumes physical storage space within the database. Some of the more common segment types are described in Table 1.7.

TABLE 1.7 Oracle Segment Types

Segment Type	Description
Table	Stores data in column and row structure.
Index	Improves the access to table data.
Rollback	Special segment used to maintain read consistency during user transactions and perform transaction recovery. Rollback segments are described in Chapter 8, "Managing Consistency and Concurrency."
Partition	Divides a table into smaller, more manageable pieces for performance purposes.

Each Oracle segment is made up of contiguous chunks of storage space in the database called *extents*. Every segment must have at least one extent, but can have as many as 2 billion extents.

NOTE Any segment whose maximum number of extents is specified as "unlimited" actually has a maximum of 2 billion extents.

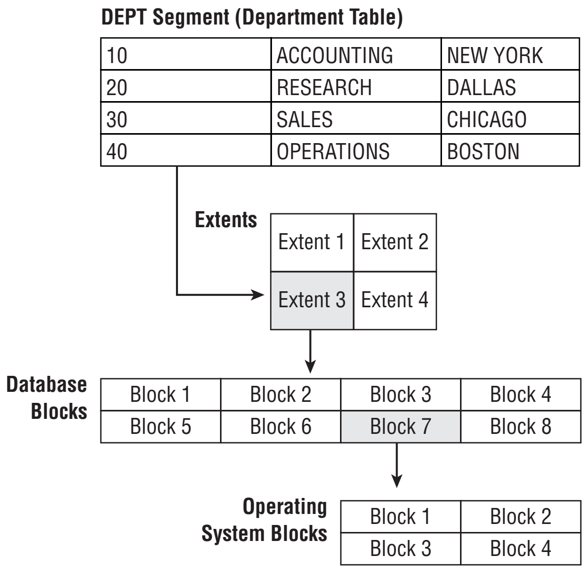
Each extent is itself made up of a collection of smaller chunks of space called Oracle *database blocks*. The minimum size of an extent is five database blocks. The default size of these database blocks is set at database creation, but Oracle 10*g* databases can use multiple block sizes within one database. The common database block sizes are 2KB, 4KB, 8KB, and 16KB.

Each database block is in turn composed of one or more *operating system blocks*. The size of an operating system block depends on the operating system, but most are 512 bytes to 2KB in size. Figure 1.4 summarizes the relationship between the segments, extents, database blocks, and operating system blocks.

Figure 1.4 illustrates how the DEPT table is made up of four extents. Each of these extents is made up of eight database blocks, and each database block is made up of four operating system blocks.

Once a segment such as a table is created, SQL is used to interact with it. The ways in which SQL accesses tables are described in the following section.

FIGURE 1.4 Segment space hierarchy



Interacting with Segments

The most common way to interact with an Oracle database is through the use of SQL. The SQL statements might be typed within an Oracle query tool, dynamically generated using a web-based development or management tool, or entered using a programming language such as C++ or COBOL. You can also use Oracle's own procedural language to extend the functionality of SQL within the database. Each of these methods of interacting with the database is explained in the following sections.

Structured Query Language

In their simplest form, SQL statements can be constructed using either of Oracle's command-line SQL tools: SQL*Plus or *i*SQL*Plus. Both SQL*Plus and *i*SQL*Plus allow you to type SQL commands and pass them directly to the database for processing. As described in the previous section, there are four types of SQL commands:

- Queries created using `SELECT` statements
- DML commands created using `INSERT`, `UPDATE`, and `DELETE` statements
- DDL commands created using `CREATE`, `ALTER`, or `DROP` commands
- DCL commands created using `GRANT` and `REVOKE` commands

NOTE Whenever DML commands are performed on a table, the rows impacted by the change are locked by Oracle. Locking is described in detail in Chapter 8.

Short examples of each of these types of SQL statements will appear throughout the remainder of this chapter. However, most of the examples are of `SELECT` statements. SQL `SELECT` statements can be composed of many parts:

- The `SELECT` list, in which each of the columns you want to include in your output is specified. The `SELECT` clause is required in all queries. An * can be used in the `SELECT` clause if every column in a table is to be included in the query output.
- The `FROM` clause in which the name of the table or tables being queried is specified. The `FROM` clause is required in all queries.
- The `WHERE` clause in which the output of the query is further restricted by placing conditions on the rows that will be returned. The `WHERE` clause is optional. If it is not used, the query returns all rows from the table. When a query joins two or more tables, you can use the `WHERE` clause to define the `JOIN` condition.
- The `GROUP BY` clause, which allows you to group related rows of data to summarize their results.
- The `HAVING` clause, which, like the `WHERE` clause, is used to reduce the output of the query by limiting which rows are returned.
- The `ORDER BY` clause that sorts the query output in a specified order.

The following example shows an example of the parts of a SQL statement.

```
SQL> select dname, SUM(sal)
  2 from DEPT, EMP
  3 where DEPT.deptno=EMP.deptno
  4 group by dname
  5 having SUM(sal) > 10000
  6 order by SUM(sal);

DNAME       SUM(SAL)
----------- --------
RESEARCH       10875
SALES          10100
```

The query examples used throughout this book use each of these clauses. The following sections show how you can use SQL*Plus and *i*SQL*Plus to issue SQL statements.

Using SQL*Plus to Access a Database

To access an Oracle database using SQL*Plus, you must have the following:

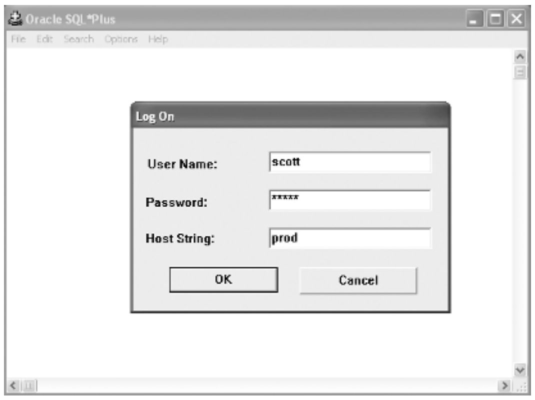
- The SQL*Plus client software on your local computer or accessible on the host server via a remote logon or Telnet session
- A valid database user name and password
- The Oracle Net connection string of the database to which you will connect

Figure 1.5 shows a user connecting with the Windows version of the SQL*Plus client to a database called PROD.

Once connected to the database via SQL*Plus, you can issue SQL statements from the SQL prompt. Figure 1.6 shows a query that returns all the columns and rows from the DEPT table.

Notice that the SELECT statement shown in Figure 1.6 ends with a semicolon (;). All SQL commands in SQL*Plus end with either a semicolon or a forward slash (/).

FIGURE 1.5 Accessing a database using SQL*Plus



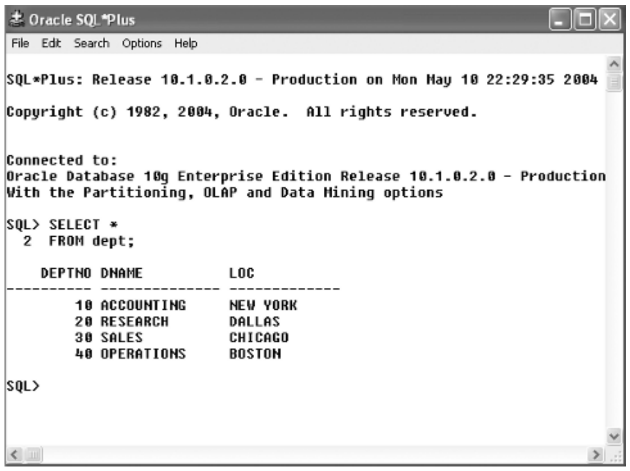
User Name: scott
Password: *****
Host String: prod

When a SQL statement ends with a forward slash (/), the forward slash should be alone on the last line of the statement.

The SQL examples in this book use mixed case to differentiate SQL reserved words from table and column names. However, SQL is not case sensitive, and you can type commands in upper, lower, or mixed case with the same results.

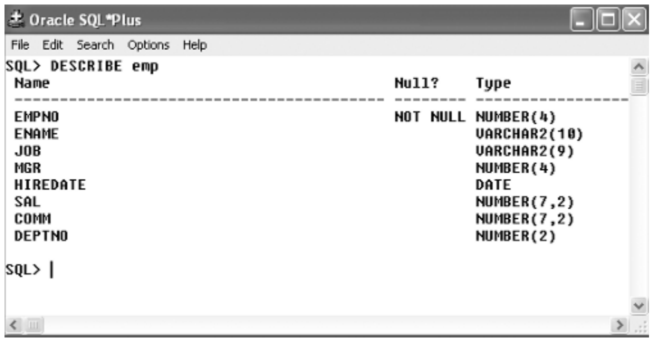
Another useful command that you can use in SQL*Plus is `DESCRIBE`. The `DESCRIBE` command is not a standard SQL command, but an Oracle-specific SQL*Plus command. It displays the logical structure of a table. Figure 1.7 shows the `DESCRIBE` command being used on the `EMP` table.

FIGURE 1.6 Querying the *DEPT* table



```
SQL*Plus: Release 10.1.0.2.0 - Production on Mon May 10 22:29:35 2004

Copyright (c) 1982, 2004, Oracle.  All rights reserved.


Connected to:
Oracle Database 10g Enterprise Edition Release 10.1.0.2.0 - Production
With the Partitioning, OLAP and Data Mining options

SQL> SELECT *
  2  FROM dept;

    DEPTNO DNAME          LOC
---------- -------------- -------------
        10 ACCOUNTING     NEW YORK
        20 RESEARCH       DALLAS
        30 SALES          CHICAGO
        40 OPERATIONS     BOSTON

SQL>
```

FIGURE 1.7 Describing the structure of the *EMP* table



```
SQL> DESCRIBE emp
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 EMPNO                                     NOT NULL NUMBER(4)
 ENAME                                              VARCHAR2(10)
 JOB                                                VARCHAR2(9)
 MGR                                                NUMBER(4)
 HIREDATE                                           DATE
 SAL                                                NUMBER(7,2)
 COMM                                               NUMBER(7,2)
 DEPTNO                                             NUMBER(2)

SQL> |
```

Using *i*SQL*Plus to Access a Database

To access an Oracle database using the browser-based *i*SQL*Plus tool, you must have the following:

- A web browser
- The URL address to the host server running the *i*SQL*Plus website
- A valid database user name and password
- The Oracle Net connection string of the database to which you will connect

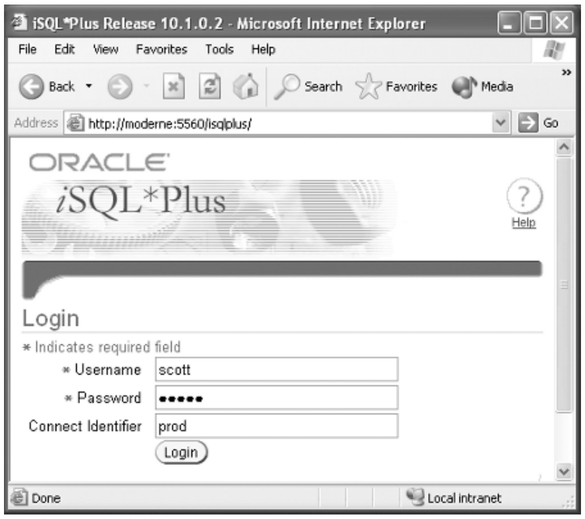
Figure 1.8 shows a user connecting the *i*SQL*Plus client to a database called PROD.

The *i*SQL*Plus interface is composed of two windows. You enter SQL statements in the top window, click the Execute button, and the output from those statements is displayed in the bottom window. Just as in SQL*Plus, each *i*SQL*Plus statement can end with a semicolon or a forward slash (/). However, *i*SQL*Plus also allows you to execute SQL commands without specifying the semicolon or forward slash. Figure 1.9 shows a `SELECT` statement that displays two of the columns and all the rows from the `EMP` table.

Web-Based Management and Development Tools

In addition to *i*SQL*Plus, Oracle provides several other web-based tools for accessing and manipulating data in databases. Most of these tools do not require that the user construct their own SQL statements the way SQL*Plus or *i*SQL*Plus do. Instead, these tools either dynamically generate SQL code or use SQL code stored in the database to interact with the database. One example of this type of query tool is Oracle Discoverer. Discoverer is an end-user query tool that allows users to run predefined and ad hoc reports from their web browser simply by clicking the tables that they want to query. Oracle Forms and Reports also allows users to access databases using web-based forms and reports.

FIGURE 1.8 Accessing a database using *i*SQL*Plus


iSQL*Plus Release 10.1.0.2 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Address
Go
ORACLE
iSQL*Plus
Help
Login
Done
Local intranet

http://moderne:5560/isqlplus/

* Indicates required field

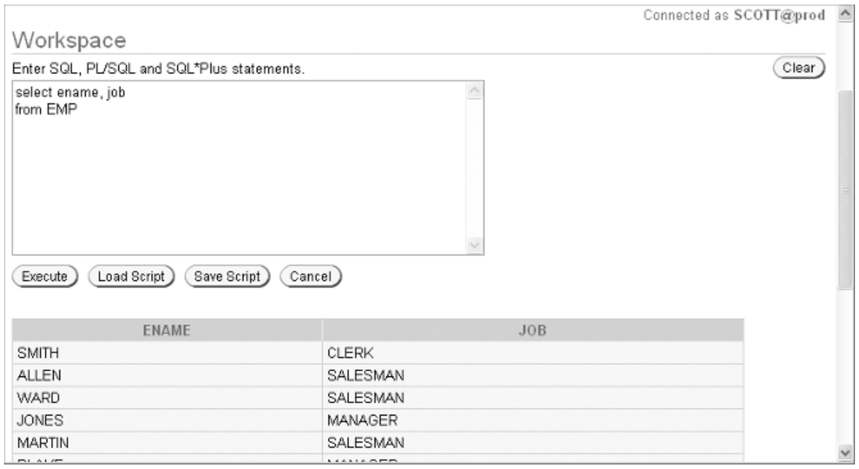
* Username	scott
* Password	●●●●●
Connect Identifier	prod

Login

NOTE The *i*SQL*Plus listener process must be running on the host server before you can connect using your web browser. You can use the operating system command `isqlplusctl start` to start the *i*SQL*Plus listener on the host server.

Another tool, *Enterprise Manager (EM) Database Control*, is Oracle's web-based database administration tool. EM Database Control dynamically produces SQL commands that are sent to the database based on the navigational choices that are made within EM Database Control. A portion of the Administration page of EM Database Control is shown in Figure 1.10.

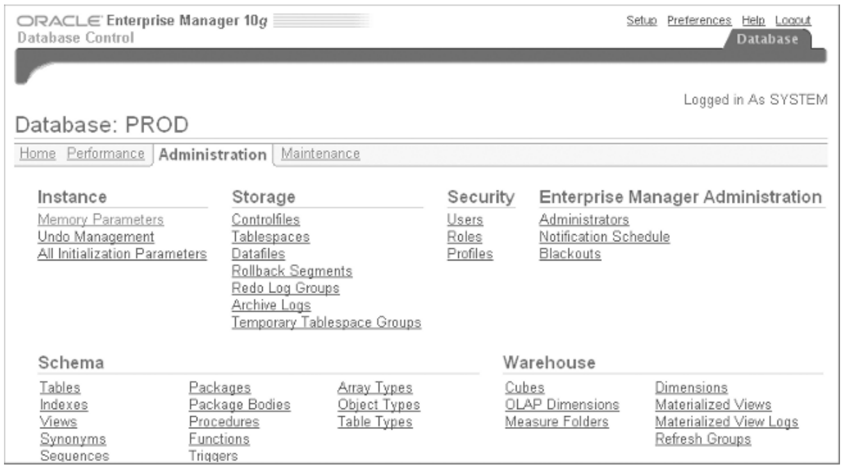
FIGURE 1.9 Querying the *EMP* table



```
select ename, job
from EMP
```

ENAME	JOB
SMITH	CLERK
ALLEN	SALESMAN
WARD	SALESMAN
JONES	MANAGER
MARTIN	SALESMAN

FIGURE 1.10 The EM Database Control main page



Most EM Database Control pages have a Show SQL button. Clicking this button displays the full text of any SQL statements that have been generated as a result of the user's actions within EM Database Control. In this manner, you can use the Show SQL button to review the SQL statements that EM sends to the database when the user clicks the OK button for the current operation.

NOTE

The EM Database Control process must be running on the host server before you can connect using your web browser. You can use the operating system command `emctl start dbconsole` to start the EM process on the host server. The default URL for accessing EM Database Control is `http://hostname:5500/em`.

In addition to EM Database Control, you can install EM client software on your computer so that you can manage database tasks without using EM Database Control, if needed.

PL/SQL: Procedural Database Language

SQL is a powerful language for interacting with databases, but it does have some limitations. For example, SQL does not have very good mechanisms for condition testing, which would allow a SQL statement to execute if a given condition is true, but not execute if the condition is false. SQL also lacks looping capabilities, the ability to perform a specific SQL action for a specified number of times before stopping. Finally, SQL does not offer any exception-handling capabilities; all errors raised by SQL statements are returned directly to the user.

Oracle Procedural Language for SQL (PL/SQL) is the solution for all these limitations. PL/SQL is a powerful extension to SQL that not only adds condition testing, looping, and exception handling, but also allows developers to write application-specific functions, procedures, packages, and triggers. Table 1.8 describes each of these types of PL/SQL objects.

TABLE 1.8 Types of PL/SQL Objects

PL/SQL Object	Description
Anonymous Block	A block of PL/SQL code that is not stored in the database, but instead is embedded in a form, web page, or SQL script.
Procedure	A block of PL/SQL code that is stored in the database and performs a specific action.
Function	A block of PL/SQL code that is stored in the database and returns a value when called in a SQL statement.
Package	A collection of related procedures and/or functions that perform related functions.
Trigger	A block of PL/SQL code that runs whenever an INSERT, UPDATE, or DELETE activity occurs on a table. Can also be defined to run when certain database events occur.

NOTE Chapter 7, "Managing Data with SQL, PL/SQL, and Utilities," provides information on how to use PL/SQL.

Accessing the Database Using Java

Since its introduction in 1995, Java has emerged as a dominant development environment for web-based applications. The primary reason for Java's popularity is its operating system independence. Java programs can be developed on one operating system and then deployed on some other operating system without modification. This is made possible by running the Java programs in an operating system–specific engine called the *Java Virtual Machine (JVM)*. In this way, the only part of the Java architecture that is operating system–specific is the JVM—not the programs themselves.

By incorporating a JVM directly in the database, Oracle 10*g* can store and execute compiled Java code natively. This not only greatly improves the performance of Java-based applications, but also allows developers to incorporate Java code directly into PL/SQL procedures, functions, and packages.

NOTE Oracle also includes a Java-based driver, the JDBC (Java Database Connectivity) driver, for improved client-to-database Java connectivity.

Using Oracle Programming Interfaces

In addition to SQL, web-based tools, PL/SQL, and Java, Oracle also provides the ability to integrate SQL commands and database connectivity into traditional programming languages such as C, C++, and COBOL. This integration is achieved by using the Oracle precompilers and the Oracle Call Interface (OCI).

Oracle precompilers allow programmers to incorporate calls to the database directly into their program code. Precompilers are available for third-generation programming languages such as C and COBOL. The Oracle C++ Call Interface (OCCI) is used with C++ to provide full database interaction with that development environment. The OCCI provides substantial programmatic support for database security and password management, access to Oracle datatypes and object-relational features, management of distributed database transactions, and globalization features.

The Oracle Architecture

Each interface described in the previous section allows a user to interact with the database. Using these tools requires that user accounts be created in the database and connectivity to the database be in place across the network. Users must also have adequate storage capacity for the data that they insert, and they need recovery mechanisms for restoring the transactions that 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 10*g* software on the server hardware
- Creating the database itself
- Creating and managing the tables and other objects used to manage the application data
- Creating and managing database users
- Establishing reliable backup and recovery processes for the database
- Monitoring and tuning database performance

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:

- User-related processes
- Logical memory structures that are collectively called an Oracle *instance*
- Physical file structures that are collectively called a *database*

Figure 1.11 shows all the parts of an Oracle instance and database.

The architecture in Figure 1.11 may at first seem complex. However, each of these architecture components is described in more detail in the following sections, beginning with the user-related processes.

NOTE Taken together, the instance and the database are called an *Oracle Server*.

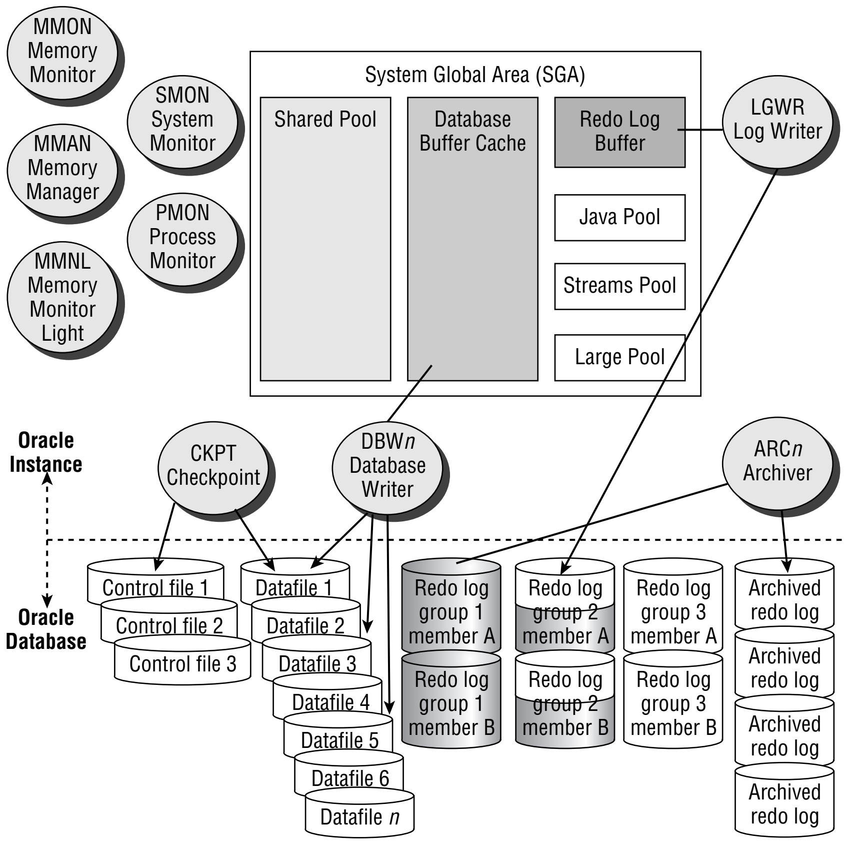
User Processes

At the user level, two 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 PC 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 then 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.

FIGURE 1.11 The Oracle 10*g* architecture

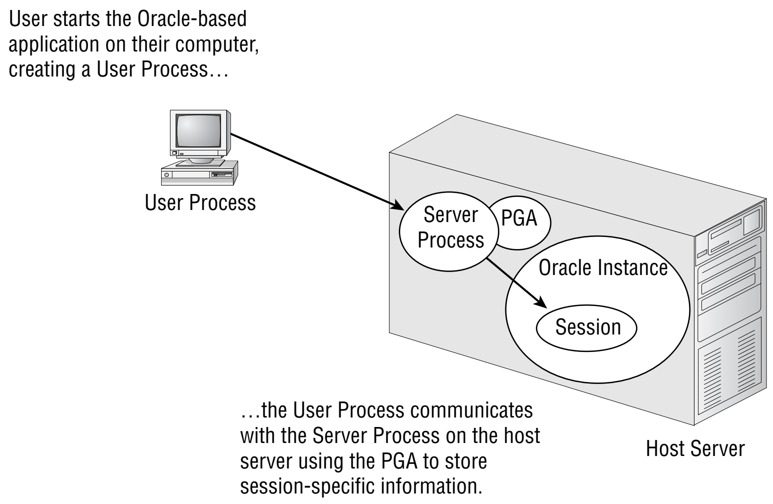


Examples of these interactions include sending SQL statements to the database, retrieving needed data from the database's physical files, and returning that data to the user.

NOTE Server Processes generally have a one-to-one relationship with User Processes—each User Process connects to one and only one Server Process. However, in some Oracle configurations, multiple User Processes can share Server Processes. See Chapter 5, "Oracle Shared Server," for details.

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 user. 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 1.12 shows the relationship between a User Process, Server Processes, PGA, and session.

FIGURE 1.12 The relationship between User and Server processes


User starts the Oracle-based application on their computer, creating a User Process…
User Process
Server Process
PGA
Oracle Instance
Session
…the User Process communicates with the Server Process on the host server using the PGA to store session-specific information.
Host Server

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 Server instance is made up of Oracle's main memory structure, called the *System Global Area (SGA)*, and several Oracle background processes. It is with the SGA that the Server Process communicates when the user accesses the data in the database. The components of the instance are described in the following sections.

The System Global Area

The SGA is made up of three required components and three optional components. Table 1.9 describes the required components, and Table 1.10 describes the optional components.

TABLE 1.9 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

TABLE 1.10 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

Oracle uses a *least recently used (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 sizes of these SGA components can be managed in two ways: manually and 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.

Whether size is managed manually or automatically, Oracle accomplishes this dynamic allocation of space within the SGA by dividing the allocated SGA memory into chunks called *granules*. These granules of memory are dynamically allocated or deallocated from the Buffer Cache, Shared Pool, Large Pool, and Java Pool as needed according to the demands placed on these areas by the application users.

NOTE Depending on your server operating system and the size of the SGA, granules can be 4MB, 8MB, or 16MB in size.

Whether the instance operates in manual or automatic mode is determined by settings in a configuration file called the parameter initialization file. There are two types of parameter initialization files: *Parameter Files (PFILES)*, and *Server Parameter Files (SPFILES)*. You can use either type of file to configure instance and database options, including the size of the SGA and its components if manual SGA management is being used, or the overall memory allocated to the SGA if automatic SGA management is being used. However, there are some important differences between the two types of configuration files, as shown in Table 1.11.

TABLE 1.11 Comparison of PFILES and SPFILES

PFILE	SPFILE
Text file that can be edited using a text editor.	Binary file that cannot be edited directly.
When changes are made to the PFILE, the instance must be shut down and restarted before it takes effect.	Most changes to the SPFILE can be made dynamically, while the instance is open and running.
Is called `init`*`instance name`*`.ora`.	Is called `spfile`*`instance name`*`.ora`.
Can be created from an SPFILE using the `create pfile from spfile` command.	Can be created from a PFILE using the `create spfile from pfile` command.

NOTE The use of automatic SGA management features requires the use of the `SPFILE` for maximum benefit.

NOTE See the section "OFA Directory Paths" later in this chapter for details on the default locations of PFILES and SPFILES.

You can specify more than 250 documented configuration parameters in the PFILE or SPFILE. Oracle 10*g* divides these parameters into two categories: basic and advanced. Oracle recommends that you set only about 30 basic initialization parameters manually. Oracle also recommends that you do not modify the remaining 220 or so parameters unless directed to do so by Oracle Support or to meet the specific needs of your application. The basic initialization parameters are described in Table 1.12.

TABLE 1.12 Oracle 10*g* Basic Initialization Parameters

Parameter Name	Description
`CLUSTER_DATABASE`	Tells the instance whether it is part of a clustered environment.
`COMPATIBLE`	Specifies the release level and feature set that you want to be active in the instance.
`CONTROL_FILES`	Designates the physical location of the database control files.
`DB_BLOCK_SIZE`	Specifies the default database block size.

TABLE 1.12 Oracle 10*g* Basic Initialization Parameters *(continued)*

Parameter Name	Description
DB_CREATE_FILE_DEST	Specifies the directory location where database datafiles will be created if the Oracle Managed Files feature is used.
DB_CREATE_ONLINE_LOG_DEST_*n*	Specifies the location(s) where the database redo log files will be created if the Oracle Managed Files feature is used.
DB_DOMAIN	Specifies the logical location of the database on the network.
DB_NAME	Specifies the name of the database that is mounted by the instance.
DB_RECOVERY_FILE_DEST	Specifies the location where recovery files will be written if the Flash Recovery feature is used.
DB_RECOVERY_FILE_DEST_SIZE	Specifies the amount of disk space available for storing Flash Recovery files.
DB_UNIQUE_NAME	Specifies a globally unique name for the database within the enterprise.
INSTANCE_NUMBER	Identifies the instance in a Real Application Clusters (RAC) environment.
JOB_QUEUE_PROCESSES	Specifies the number of background processes to start for handling jobs submitted via Enterprise Manager or DBMS_JOBS.
LOG_ARCHIVE_DEST_*n*	Specifies as many as nine locations where archived redo log files are to be written.
LOG_ARCHIVE_DEST_STATE_*n*	Indicates how the specified locations should be used for log archiving.
NLS_LANGUAGE	Specifies the default language of the database.
NLS_TERRITORY	Specifies the default region or territory of the database.
OPEN_CURSORS	Sets the maximum number of cursors that an individual session can have open at one time.
PGA_AGGREGATE_TARGET	Establishes the overall amount of memory that all PGA processes are allowed to consume.
PROCESSES	Specifies the maximum number of operating system processes that can connect to the instance.

TABLE 1.12 Oracle 10*g* Basic Initialization Parameters *(continued)*

Parameter Name	Description
REMOTE_LISTENER	Specifies a network name that points to the address or list of addresses of remote Oracle Net listeners.
REMOTE_LOGIN_PASSWORDFILE	Determines whether the instance uses a password file and what type.
ROLLBACK_SEGMENTS	Specifies only if Automatic Undo Management is not being used.
SESSIONS	Determines the maximum number of sessions that can connect to the database.
SGA_TARGET	Establishes the maximum size of the SGA, within which space is automatically allocated to each SGA component when automatic memory management is used.
SHARED_SERVERS	Specifies the number of Shared Server processes to start when the instance is started. See Chapter 5 for details.
STAR_TRANSFORMATION_ENABLED	Determines whether the optimizer will consider star transformations when queries are executed. See Chapter 9, "Proactive Database Maintenance and Performance Monitoring," for details on the optimizer.
UNDO_MANAGEMENT	Establishes whether system undo is automatically or manually managed. See Chapter 8 for details on undo segments.
UNDO_TABLESPACE	Specifies which tablespace stores undo segments if the Automatic Undo Management option is used. See Chapter 8 for details on undo management.

As shown in Table 1.12, many initialization parameters are used to specify the size of the SGA and its components. Any parameters not specified in the PFILE or SPFILE take on their default values. The following is an example of the contents of a typical Unix Oracle 10*g* PFILE that contains both basic and advanced parameters:

```
db_block_size=8192
db_file_multiblock_read_count=16
open_cursors=300
db_name=PROD
background_dump_dest=/u01/app/oracle/admin/PROD/bdump
```

```
core_dump_dest=/u01/app/oracle/admin/PROD/cdump
user_dump_dest=/u01/app/oracle/admin/PROD/udump
control_files=(/u02/oradata/PROD/control01.ctl,
               /u03/oradata/PROD/control02.ctl,
               /u05/oradata/PROD/control03.ctl)
db_recovery_file_dest=/u01/app/oracle/flash_recovery_area/
db_recovery_file_dest_size=2147483648
job_queue_processes=10
compatible=10.1.0.2.0
sga_target=500M
max_sga_size=800M
processes=250
remote_login_passwordfile=EXCLUSIVE
pga_aggregate_target=25165824
sort_area_size=65536
undo_management=AUTO
undo_tablespace=UNDOTBS1
```

In this sample PFILE, the sizes of the Shared Pool, Database Buffer Cache, Large Pool, and Java Pool are not individually specified. Instead, Oracle 10*g*'s automatic memory management features allow you to simply set one configuration parameter—`SGA_TARGET`—to establish the total amount of memory allocated to the SGA. Oracle then automatically allocates portions of this overall memory allocation to each of the SGA components at instance startup and also dynamically reallocates the space as needed to maximize performance while the database is in use. In addition to examining the PFILE/SPFILE, you can also use the `V$SGA` and `V$SGA_DYNAMIC_COMPONENTS` dynamic performance view to display the size of the SGA and some of its components, as shown here:

```
SQL> select *
  2  from V$SGA;

NAME                      VALUE
-------------------- ----------
Fixed Size               787988
Variable Size         145750508
Database Buffers       25165824
Redo Buffers             262144
```

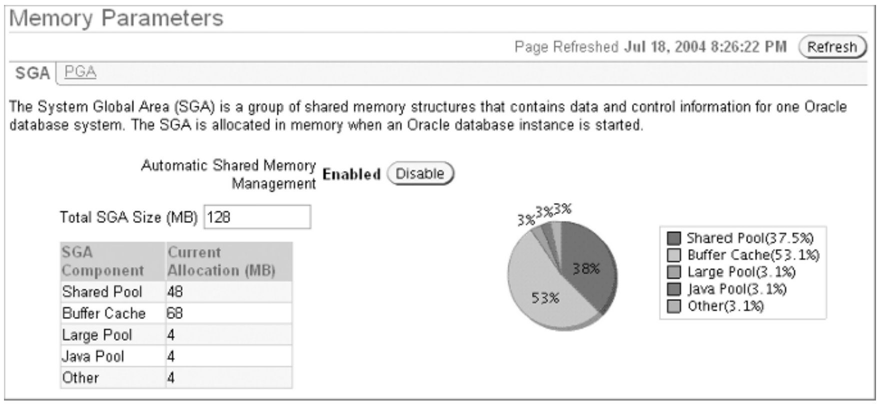
The output from this query shows that the total size of the SGA is 171,966,464 bytes. This total size is composed of the variable space that is composed of the Shared Pool, the Large Pool, and the Java Pool (145,750,508 bytes), the Database Buffer Cache (25,165,824 bytes), the Redo Log Buffer (262,144 bytes), and some additional space (787,988 bytes) that stores information used by the instance's background processes. The `V$SGA_DYNAMIC_COMPONENTS` view displays additional detail about the allocation of space within the SGA, as shown in the following query:

```
SQL> select component,current_size
  2  from v$sga_dynamic_components;

COMPONENT                                CURRENT_SIZE
---------------------------------------- ------------
shared pool                                  83886080
large pool                                    8388608
java pool                                    50331648
streams pool                                        0
DEFAULT buffer cache                         25165824
KEEP buffer cache                                   0
RECYCLE buffer cache                                0
DEFAULT 2K buffer cache                             0
DEFAULT 4K buffer cache                             0
DEFAULT 8K buffer cache                             0
DEFAULT 16K buffer cache                            0
DEFAULT 32K buffer cache                            0
OSM Buffer Cache                                    0

13 rows selected.
```

The results of this query show that the Shared Pool is 83,886,080 bytes, the Large Pool is 8,388,608 bytes, the Java Pool is 50,331,648 bytes, and the Database Buffer Cache is 25,165,824 bytes.

You can also use EM Database Control to view the sizes of each of the SGA components, as shown in Figure 1.13.

FIGURE 1.13 EM Database Control showing SGA components



Memory Parameters

Page Refreshed **Jul 18, 2004 8:26:22 PM**

The System Global Area (SGA) is a group of shared memory structures that contains data and control information for one Oracle database system. The SGA is allocated in memory when an Oracle database instance is started.

Automatic Shared Memory Management **Enabled**

Total SGA Size (MB) 128

SGA Component	Current Allocation (MB)
Shared Pool	48
Buffer Cache	68
Large Pool	4
Java Pool	4
Other	4

Real World Scenario

Handle With Care: Undocumented Configuration Parameters

You've just read a performance-tuning tip posted to the Oracle newsgroup at `comp.databases.oracle.server`. The person posting the tip suggests setting the undocumented PFILE parameter `_dyn_sel_est_num_blocks` to a value of 200 in order to boost your database's performance. Should you implement this suggestion?

More than 1000 undocumented configuration parameters are available in Oracle 10*g*. Undocumented configuration parameters are distinguished from their documented counterparts by the underscore that precedes their name, as with the parameter described in the newsgroup posting.

I do not recommend utilizing undocumented PFILE or SPFILE parameters on any of your systems because knowing the appropriate reasons to use these parameters, and the appropriate values to set these parameters to, is almost pure speculation because of their undocumented nature. Although some undocumented parameters are relatively harmless, using others incorrectly can cause unforeseen database problems. What does the `_dyn_sel_est_num_blocks` parameter do, and what value should you set it to? Only the engineers of the Oracle 10*g* kernel code know for sure.

One exception to this suggestion is when you are directed to use an undocumented configuration parameter by Oracle Support. Oracle Support occasionally uses these parameters to enhance the generation of debug information or to work around a bug in the kernel code.

Oracle Background Processes

There are many types of Oracle background processes. Each performs a specific job in helping to manage the instance. Five Oracle background processes are required, 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 which optional Oracle features are being used in the database. Table 1.13 describes the required background processes, and Table 1.14 describes some of the optional background processes.

TABLE 1.13 Required Oracle Background Processes

Process Name	Operating System Process	Description
System Monitor	`SMON`	Performs instance recovery following an instance crash, coalesces free space in the database, and manages space used for sorting
Process Monitor	`PMON`	Cleans up failed user database connections

TABLE 1.13 Required Oracle Background Processes *(continued)*

Process Name	Operating System Process	Description
Database Writer	`DBW`*n**	Writes modified database blocks from the SGA's Database Buffer Cache to the datafiles on disk
Log Writer	`LGWR`	Writes transaction recovery information from the SGA's Redo Log Buffer to the online Redo Log files on disk
Checkpoint	`CKPT`	Updates the database files following a Checkpoint Event

*The *n* in any operating system process name signifies that more than one of these processes may be running. In these cases, the n is replaced with a numeric value. For example, if four database writer processes are running, their process names at the operating system level are DBW0, DBW1, DBW2, and DBW3.

TABLE 1.14 Optional Oracle Background Processes

Process Name	Operating System Process	Description
Archiver	`ARC`*n*	Copies the transaction recovery information written to disk by LGWR (log writer) to the online Redo Log files and to a secondary location in case it is needed for recovery. Nearly all production databases use this optional process. See Chapter 2, "Creating and Controlling a Database," for details on how to enable this process.
Recoverer	`RECO`	Recovers failed transactions that are distributed across multiple databases when using Oracle's distributed database feature.
Job Queue Monitor	`CJQ`*n*	Assigns jobs to the Job Queue processes when using Oracle's job scheduling feature.
Job Queue	`J`*nnn*	Executes database jobs that have been scheduled using Oracle's job scheduling feature.
Queue Monitor	`QMN`*n*	Monitors the messages in the message queue when Oracle's Advanced Queuing feature is used.
Parallel Query Slave	`Q`*nnn*	Used to carry out portions of a larger overall query when Oracle's Parallel Query feature is used.

TABLE 1.14 Optional Oracle Background Processes *(continued)*

Process Name	Operating System Process	Description
Dispatcher	D*nnn*	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 5 for details on using Shared Servers.
Shared Server	S*nnn*	Server Processes that are shared among several users when Oracle's Shared Server feature is used. See Chapter 5 for details on using Shared Servers.
Memory Manager	`MMAN`	Manages the size of each individual SGA component when Oracle's Automatic Shared Memory Management feature is used. See Chapter 9 for more information on using this feature.
Memory Monitor	`MMON`	Gathers and analyzes statistics used by the Automatic Workload Repository feature. See Chapter 9 for more information on using this feature.
Memory Monitor Light	`MMNL`	Gathers and analyzes statistics used by the Automatic Workload Repository feature. See Chapter 9 for more information on using this feature.
Recovery Writer	`RVWR`	Writes recovery information to disk when Oracle's Flashback Database Recovery feature is used. See Chapter 10, "Implementing Database Backups," and Chapter 11, "Implementing Database Recovery," for details on how to use the Flashback Database Recovery feature.
Change Tracking Writer	`CTWR`	Keeps track of which database blocks have changed when Oracle's incremental Recovery Manager feature is used. See Chapters 10 and 11 for details on using Recovery Manager to perform backups.

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

```
$ ps -ef |grep PROD
oracle    3969     1  0 10:02 ?        00:00:05 ora_pmon_PROD
oracle    3971     1  0 10:02 ?        00:00:00 ora_mman_PROD
     oracle    3973     1  0 10:02 ?        00:00:07 ora_dbw0_PROD
     oracle    3975     1  0 10:02 ?        00:00:07 ora_lgwr_PROD
     oracle    3977     1  0 10:02 ?        00:00:10 ora_ckpt_PROD
     oracle    3979     1  0 10:02 ?        00:00:20 ora_smon_PROD
     oracle    3981     1  0 10:02 ?        00:00:00 ora_reco_PROD
     oracle    3983     1  0 10:02 ?        00:00:09 ora_cjq0_PROD
     oracle    3985     1  0 10:02 ?        00:00:00 ora_d000_PROD
     oracle    3987     1  0 10:02 ?        00:00:00 ora_s000_PROD
     oracle    4052     1  0 10:02 ?        00:00:00 ora_qmnc_PROD
     oracle    4054     1  0 10:02 ?        00:00:29 ora_mmon_PROD
     oracle    4057     1  0 10:02 ?        00:00:08 ora_mmnl_PROD
     oracle    4059     1  0 10:02 ?        00:01:04 ora_j000_PROD
     oracle   27544     1  0 20:29 ?        00:00:00 ora_q000_PROD
```

This output shows that 15 background processes are running on the Unix server for the PROD database.

NOTE User Server processes are not considered part of the instance.

NOTE In Windows environments, a Windows service called OracleService*Instance-Name* is also associated with each instance. This service must be started in order to start up the instance in Windows environments.

The Oracle Database

An instance is a temporary memory structure, but the Oracle database is made up of a set of physical files that reside on the host server's disk drives. These files are called *control files*, *datafiles*, and *redo logs*. Additional physical files that are associated with the Oracle database, but are not technically part of the database, are the *password file*, the PFILE and SPFILE described previously, and any *archived redo log files*. Table 1.15 summarizes the role that each of these files plays in the database architecture.

NOTE Creating and managing these files is discussed in detail in Chapter 2.

The three files that make up a database—the control file, datafile, and redo logs—are described in the following sections.

TABLE 1.15 Oracle Physical Files

File Type	Information Contained in File(s)
Control	Locations of other physical files, database name, database block size, database character set, and recovery information. These files are required to open the database.
Datafile	All application data and internal metadata.
Redo log	Record of all changes made to the database; used for recovery.
Parameter (PFILE or SPFILE)	Configuration parameters for the SGA, optional Oracle features, and background processes.
Archived log	Copy of the contents of previous online redo logs, used for recovery.
Password	Optional file used to store names of users who have been granted the SYSDBA and SYSOPER privileges. See Chapter 6 for details on SYSDBA and SYSOPER privileges.
Oracle Net	Entries that configure the database listener and client-to-database connectivity. See Chapter 4 for details.

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
- The names, locations, and sizes of the datafiles and redo log files
- Information used to recover the database in the case of a disk failure or user error

The control files are created when the database is created in the locations specified in the `control_files` parameter in the parameter file. Because loss of the control files negatively impacts the ability to recover the database, most production 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$CONTROLFILE` to display the names and locations of all the database's control files. A sample query of `V$CONTROLFILE` on a Unix system is shown here:

```
SQL> select name from v$controlfile;

NAME
------------------------------------
/u02/oradata/PROD/control01.ctl
```

```
/u03/oradata/PROD/control02.ctl
/u05/oradata/PROD/control03.ctl
```

This query shows that the database has three control files, called `control01.ctl`, `control02.ctl`, and `control03.ctl`, which are stored in the directories `/u02/oradata/PROD/`, `/u03/oradata/PROD/`, and `/u05/oradata/PROD/` respectively. You can also monitor control files using EM Database Control, as shown in Figure 1.14.

NOTE

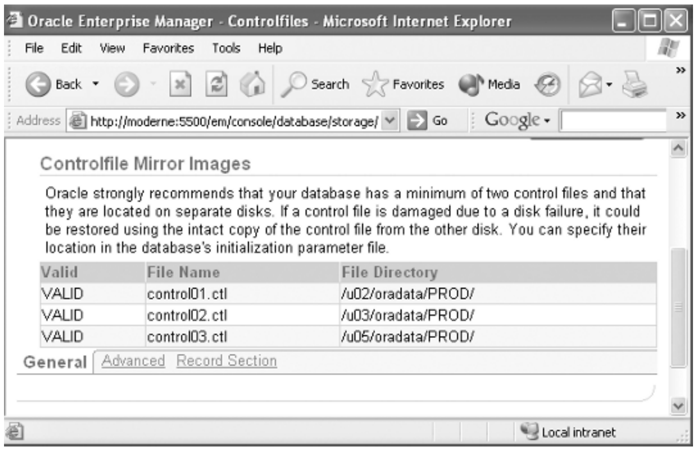
Control files are usually the smallest files in the database, generally between 1MB and 5MB in size. However, they can be larger depending on the PFILE/SPFILE setting for `CONTROLFILE_RECORD_KEEP_TIME` when the Recovery Manager feature is used.

One thing that the control files keep track of in the database are the names, locations, and sizes of the database datafiles. Datafiles, and their relationship to another database structure called a tablespace, are examined in the next section.

Datafiles

Datafiles are the physical files that actually store the data that has been inserted into each table in the database. The size of the datafiles is directly related to the amount of table data that they store. Datafiles 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`.

FIGURE 1.14 EM Database Control showing control files


Oracle Enterprise Manager - Controlfiles - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://moderne:5500/em/console/database/storage/ Go
Google
General Advanced Record Section
Local intranet

Controlfile Mirror Images

Oracle strongly recommends that your database has a minimum of two control files and that they are located on separate disks. If a control file is damaged due to a disk failure, it could be restored using the intact copy of the control file from the other disk. You can specify their location in the database's initialization parameter file.

Valid	File Name	File Directory
VALID	control01.ctl	/u02/oradata/PROD/
VALID	control02.ctl	/u03/oradata/PROD/
VALID	control03.ctl	/u05/oradata/PROD/

By default, every Oracle 10*g* database must have at least three tablespaces. These tablespaces are described in Table 1.16.

TABLE 1.16 Required Tablespaces in Oracle 10*g*

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 3 for details.

In addition to these three required tablespaces, most databases have tablespaces for storing other database segments similar to those shown in Table 1.17.

TABLE 1.17 Common Tablespaces in Oracle 10*g* Databases

Tablespace Name	Description
TOOLS	Used to store segments for nonapplication management tools.
USERS	Used as the default tablespace for database users.
UNDOTBS1	Used to store transaction information for read consistency and recovery purposes. See Chapter 8 for details.

Beyond the six common tablespaces listed in Tables 1.16 and 1.17, 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. You can use the DBA_TABLESPACES data dictionary view to display a list of all the tablespaces in the database. This is a sample query on DBA_TABLESPACES:

```
SQL> select tablespace_name
  2  from dba_tablespaces
  3  order by tablespace_name;

TABLESPACE_NAME
------------------------------
APPL_IDX
```

```
APPL_TAB
EXAMPLE
SYSAUX
SYSTEM
TEMP
UNDOTBS1

7 rows selected.
```

This output shows that this database consists of seven tablespaces: SYSTEM, UNDOTBS1, SYSAUX, TEMP, EXAMPLE, APPL_TAB, and APPL_IDX. You can also monitor tablespaces using EM Database Control, as shown in Figure 1.15.

NOTE The current release of Oracle's Application 11*i* uses more than 375 tablespaces to store application data and indexes.

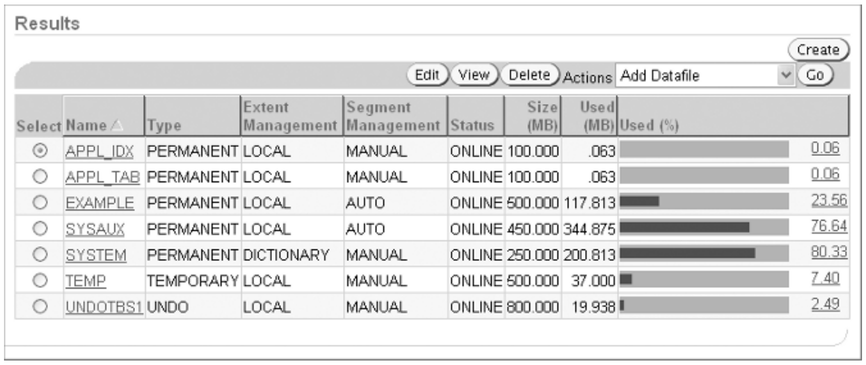
For each of the tablespaces shown in Figure 1.15, there must be at least one datafile. Some tablespaces may be composed of several datafiles for management or performance reasons. The data dictionary view DBA_DATA_FILES shows the datafiles associated with each tablespace in the database. The following 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;

TABLESPACE_NAME                FILE_NAME
------------------------------ -----------------------------------
APPL_IDX                       /u01/oradata/PROD/appl_idx01.dbf
APPL_IDX                       /u03/oradata/PROD/appl_idx02.dbf
APPL_TAB                       /u01/oradata/PROD/appl_tab01.dbf
APPL_TAB                       /u03/oradata/PROD/appl_tab02.dbf
EXAMPLE                        /u02/oradata/PROD/example01.dbf
SYSAUX                         /u04/oradata/PROD/sysaux01.dbf
SYSTEM                         /u05/oradata/PROD/system01.dbf
UNDOTBS1                       /u02/oradata/PROD/undotbs101.dbf
```

This output shows that the datafiles for the six tablespaces in this database are stored in the /u01/oradata/PROD through /u05/oradata/PROD directories. The APPL_DATA and APPL_INDX tablespaces are each made up of two datafiles; the rest of the tablespaces are each made up of only one datafile. You can also monitor datafiles using EM Database Control, as shown in Figure 1.16.

NOTE Temporary tablespaces are not displayed in DBA_TABLESPACES. They are listed in DBA_TEMP_FILES.

FIGURE 1.15 EM Database Control showing tablespaces



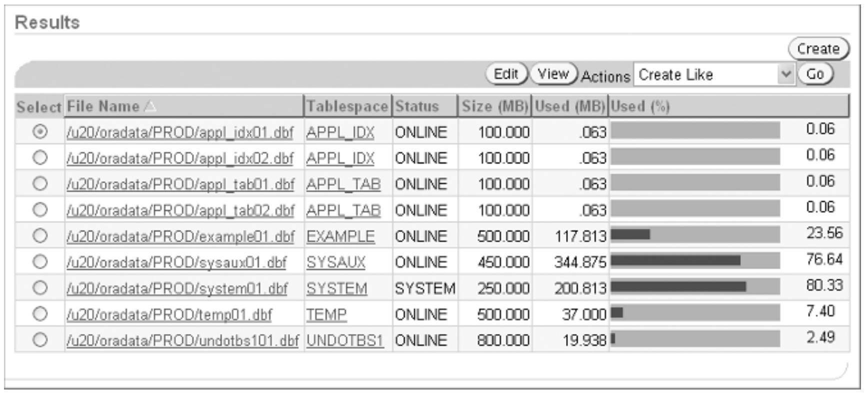
Results

Create

Edit View Delete Actions Add Datafile Go

Select	Name	Type	Extent Management	Segment Management	Status	Size (MB)	Used (MB)	Used (%)
◉	APPL_IDX	PERMANENT	LOCAL	MANUAL	ONLINE	100.000	.063	0.06
○	APPL_TAB	PERMANENT	LOCAL	MANUAL	ONLINE	100.000	.063	0.06
○	EXAMPLE	PERMANENT	LOCAL	AUTO	ONLINE	500.000	117.813	23.56
○	SYSAUX	PERMANENT	LOCAL	AUTO	ONLINE	450.000	344.875	76.64
○	SYSTEM	PERMANENT	DICTIONARY	MANUAL	ONLINE	250.000	200.813	80.33
○	TEMP	TEMPORARY	LOCAL	MANUAL	ONLINE	500.000	37.000	7.40
○	UNDOTBS1	UNDO	LOCAL	MANUAL	ONLINE	800.000	19.938	2.49

FIGURE 1.16 EM Database Control showing datafiles



Results

Create

Edit View Actions Create Like Go

Select	File Name	Tablespace	Status	Size (MB)	Used (MB)	Used (%)
◉	/u20/oradata/PROD/appl_idx01.dbf	APPL_IDX	ONLINE	100.000	.063	0.06
○	/u20/oradata/PROD/appl_idx02.dbf	APPL_IDX	ONLINE	100.000	.063	0.06
○	/u20/oradata/PROD/appl_tab01.dbf	APPL_TAB	ONLINE	100.000	.063	0.06
○	/u20/oradata/PROD/appl_tab02.dbf	APPL_TAB	ONLINE	100.000	.063	0.06
○	/u20/oradata/PROD/example01.dbf	EXAMPLE	ONLINE	500.000	117.813	23.56
○	/u20/oradata/PROD/sysaux01.dbf	SYSAUX	ONLINE	450.000	344.875	76.64
○	/u20/oradata/PROD/system01.dbf	SYSTEM	SYSTEM	250.000	200.813	80.33
○	/u20/oradata/PROD/temp01.dbf	TEMP	ONLINE	500.000	37.000	7.40
○	/u20/oradata/PROD/undotbs101.dbf	UNDOTBS1	ONLINE	800.000	19.938	2.49

NOTE Chapter 3 discusses the creation and management of tablespaces in further detail.

NOTE Datafiles 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 datafiles into the Database Buffer Cache in the SGA. If the user has performed a committed transaction that modifies that data, the Database Writer process (DBW*n*) ultimately writes the modified data back to the datafiles.

When Does Database Writer Write?

The `DBWn` background process writes to the datafiles 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 Chapters 10 and 11 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 changed to `READ ONLY`.
- A segment is dropped.

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 automatically recorded in the Redo Log Buffer. The contents of the Redo Log Buffer are ultimately written to the redo logs by the LGWR background process.

Because of the important role that redo logs play in Oracle's recovery mechanism, they are usually multiplexed, or copied. This means that each redo log contains one or more copies of itself in case one of the copies becomes corrupt or is lost due to 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, as shown in Figure 1.17.

Figure 1.17 shows a database that has three redo log groups: group 1, group 2, and group 3. Each group is composed of two members. The first member of each group is stored in the directory `/u02/oradata/PROD`. The second, multiplexed member is stored in the directory `/u04/oradata/PROD`. 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 next. The following SQL statement shows a sample query on a Unix system of the `DBA_DATA_FILES` data dictionary view.

```
SQL> select group#, member
  2  from v$logfile
  3  order by group#;

    GROUP# MEMBER
---------- ------------------------------
         1 /u02/oradata/PROD/redo01a.rdo
```

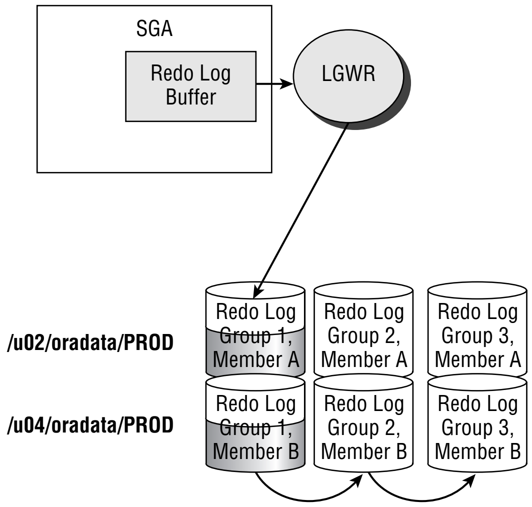
```
          1 /u04/oradata/PROD/redo01b.rdo
          2 /u02/oradata/PROD/redo02a.rdo
          2 /u04/oradata/PROD/redo02b.rdo
          3 /u02/oradata/PROD/redo03a.rdo
          3 /u04/oradata/PROD/redo03b.rdo

6 rows selected.
```

When Does Log Writer Write?

The LGWR background process writes to the current redo log group whenever one of the following database events occurs:

- Every three seconds.
- A user commits a transaction.
- The Redo Log Buffer is one-third full.
- The Redo Log Buffer contains 1MB worth of redo information.
- Before the `DBWn` *process* whenever a database checkpoint occurs. See Chapter 10 for more information on checkpoints.

FIGURE 1.17 How redo logs are used in the database


SGA
Redo Log Buffer
LGWR
/u02/oradata/PROD
/u04/oradata/PROD
Redo Log Group 1, Member A
Redo Log Group 2, Member A
Redo Log Group 3, Member A
Redo Log Group 1, Member B
Redo Log Group 2, Member B
Redo Log Group 3, Member B

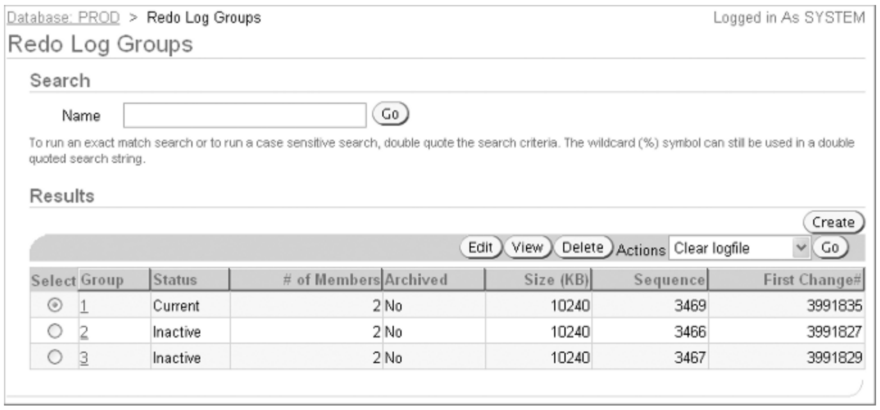
This output shows that the database has a total of three redo log groups and that each group has two members. Each of the members is located in a separate directory and disk volume on the server's disk drives so that the loss of a single disk drive will not cause the loss of the recovery information stored in the redo logs. You can also monitor redo logs using EM Database Control, as shown in Figure 1.18.

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
```

This output shows that redo log group number 1 is currently active and being written to by LGWR. Once redo log group 1 is full, LGWR switches to redo log group 2.

FIGURE 1.18 EM Database Control showing redo logs



Database: PROD > Redo Log Groups   Logged in As SYSTEM

Redo Log Groups

Search

Name

To run an exact match search or to run a case sensitive search, double quote the search criteria. The wildcard (%) symbol can still be used in a double quoted search string.

Results

Actions: Clear logfile

Select	Group	Status	# of Members	Archived	Size (KB)	Sequence	First Change#
●	1	Current	2	No	10240	3469	3991835
○	2	Inactive	2	No	10240	3466	3991827
○	3	Inactive	2	No	10240	3467	3991829

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 ARC*n* 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. This process is shown graphically in Figure 1.19.

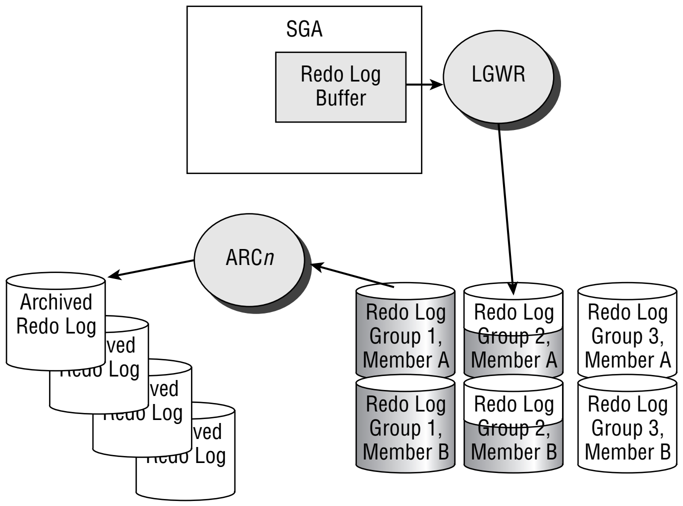
In Figure 1.19, the first redo log group has been filled, and LGWR has moved to on to redo log group 2. As soon as LGWR switches from redo log group 1 to redo log group 2, the ARC*n* process starts copying the contents of redo log group 1 to the archive log file location. Once the first redo log group is safely archived, LGWR is free to wrap around and reuse the first redo log group once redo log group 3 is filled.

NOTE 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.

NOTE 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.

The next section talks about how to install and configure the Oracle software on your server so that you can then create a database. Creating a database is described in detail in Chapter 2.

FIGURE 1.19 How *ARCn* copies redo log entries to disk


SGA
Redo Log Buffer
LGWR
ARCn
Archived Redo Log
Redo Log Group 1, Member A
Redo Log Group 2, Member A
Redo Log Group 3, Member A
Redo Log Group 1, Member B
Redo Log Group 2, Member B
Redo Log Group 3, Member B

Installing Oracle 10*g*

One of your duties as an Oracle DBA is to install and configure the Oracle 10*g* software on the server so that a database can be created to store application data. This section discusses each of the steps that you must perform in order to successfully install Oracle 10*g*.

NOTE The examples in this section are for a Unix server, but most of the concepts apply equally to Windows platforms. Significant differences between Unix and Windows are noted.

Review the Documentation

Before beginning an installation of Oracle 10*g*, you need to review several documents so that you completely understand the installation requirements. These documents include the following:

- The installation guide for your operating system
  - *Oracle Database Installation Guide 10g* Release 1 (10.1) for Unix Systems: AIX-Based Systems, HP-UX, HP Tru64 Unix, Linux, and Solaris Operating System (SPARC), Part No. B10811-02
  - *Oracle Database Installation Guide 10g* Release 1 (10.1.0.2.0) for Windows, Part No. B10130-01
- The general release notes for the version of Oracle that you are installing
- The operating system–specific release notes for the version of Oracle that you are installing
- Any "quick start" installation guides

Before you begin, review each of these documents so that you are thoroughly familiar with the install process and any known associated issues.

NOTE All these documents are available on Oracle's Technology Network website located at `http://otn.oracle.com/index.html` and on Oracle's MetaLink website at `http://metalink.oracle.com`. Unlike MetaLink accounts, OTN user accounts are free.

Review the System Requirements

The next task is to review your server hardware specifications to see if they meet or exceed the specifications in the install documentation. Minimally, this means that you must confirm that your server meets the installation requirements in these four areas:

- The operating system is of the proper release level
- The server has adequate memory to perform the install and run an instance
- The server has adequate CPU resources to perform the install and run an instance
- The server has adequate disk storage space to perform the install and run a database

4367c01.fm Page 49 Wednesday, April 6, 2005 8:14 AM

The recommended minimum hardware requirements for an Oracle 10*g* installation are shown in Table 1.18.

TABLE 1.18 Recommended Minimum Hardware Requirements for Oracle 10*g*

Hardware Component	Recommended Requirement
Memory	512MB.
Swap space	1GB or two times the amount of RAM.
Temp space	400MB of free space in the `/tmp` directory on Unix systems.
Free disk space	1.5GB of disk space is required for the base Oracle 10*g* installation. 1GB of disk space is needed to create a database using the Database Configuration Assistant.

The Oracle Universal Installer, which is described in the subsequent section, "Using the Oracle Universal Installer," will perform a quick system check prior to starting an installation to see if your system meets the specific requirements for your operating system. If your system does not meet the minimum requirements, the installer returns an error and aborts.

On Unix systems, you must examine one critical system requirement before installation: Unix kernel parameters. Unix kernel parameters are used to configure the Unix operating system settings for operating system–level operations that impact Oracle-related activities such as the following:

- The maximum size allowed for a sharable memory segment on the server, which can impact the SGA size
- The maximum number of files that can be open on the server at one time, which impacts the total number of users and files in the database
- The number of processes that can run concurrently on the server, which impacts the number of users and the ability to use some optional features

The systems administrator usually makes Unix kernel changes, which may require a server reboot in order to take effect. The install guide and/or release notes provides details on the appropriate kernel setting for your operating system. In addition to kernel settings, the system administrator may have to configure the server's disk storage system and backup hardware before installing the Oracle software.

Plan Your Install

Once you review the documentation and system requirements, you are ready to begin planning your installation. This is the last step before actually running the Oracle Universal Installer.

One way to simplify installation planning is to adopt the *Optimal Flexible Architecture (OFA)* model that Oracle recommends as a best-practice methodology for managing Oracle installations in Unix environments (and to a lesser extent, Windows environments). Cary Millsap designed the OFA model to produce database installations that are easier to manage, upgrade, and back up while at the same time minimizing problems associated with database growth. The OFA model addresses four areas:

- Naming conventions for Unix file systems and mount points
- Naming conventions for directory paths
- Naming conventions for database files
- Standardized locations for Oracle-related files

NOTE You can download Cary Millsap's original 1995 OFA white paper at `www.hotsos.com/downloads/visitor/00000019.pdf`.

In addition to using the OFA model, planning your install also means answering the following questions:

- Which operating system user will own the installed Oracle software?
- On which disk drive and directory will the Oracle software be installed?
- What directory structure will be used to store the Oracle software, its related configuration files, and the database itself?
- How should the database files be laid out so that the maximum performance benefits will be realized?
- How should the database files be laid out so that the maximum recoverability benefits will be realized?

Creating the Oracle User Account

On Unix systems, every file is owned by an operating system user account. Therefore, before you can install the Oracle software, you must create a Unix user account that will own the Oracle binaries. The user name for this account can be anything, but common Oracle user names include `oracle`, `ora10g`, and `ora101`. Each Unix user is also in one or more operating system groups. Create a new operating system group for the Oracle Unix user. This group is usually called `dba`, and you will be prompted for it later during the installation.

Naming Volumes and Mount Points

Unless Oracle's automatic storage management feature or raw devices are used, almost all files on a Unix server are stored on logical storage areas called volumes, which are attached, or mounted, to directories, or mount points, by the Unix system administrator. The OFA model suggests that these mount points be given a name that consists of a combination of a character and numeric values. Examples of common OFA mount points for Unix systems include the following:

- `/u01`
- `/mnt01`

- `/du01`
- `/d01`

Notice that the naming convention for these mount points is generic. The mount point's name has no relationship to what type of file it will ultimately hold. The OFA model recommends this generic naming convention because it provides the greatest flexibility for future management of the server's file systems.

NOTE The concept of mount points does not apply directly to Windows environments. Windows environments assign a standard Windows drive letter (for example, C:, D:) to each volume.

Creating OFA Directory Paths

The OFA model prescribes that the directory structures under the mount points use a consistent and meaningful naming convention. In addition to this naming convention, the OFA model also assigns standard operating system environment variable names to some of these directory paths as "nicknames" to aid in navigation and ensure portability of the directory structures in the event that they need to be moved to new file systems.

Tables 1.19 and 1.20 show the two operating system environment variables used in the OFA model, along with the directories with which the variables are associated, for both Unix and Windows systems.

TABLE 1.19 Comparison of Unix Directory Paths and Variables

Environment Variable	Directory Path	Description
`$ORACLE_BASE`	`/u01/app/oracle`	Top-level directory for Oracle software on the host server
`$ORACLE_HOME`	`/u01/app/oracle/product/10.1.0`	Directory into which the Oracle 10*g* software will be installed

TABLE 1.20 Comparison of Windows Directory Paths and Variables

Environment Variable	Directory Path	Description
`%ORACLE_BASE%`	`D:\ORACLE`	Top-level directory for Oracle software on the host server
`%ORACLE_HOME%`	`D:\ORACLE\ORA101`	Directory into which the Oracle 10*g* software will be installed

These environment variables are used extensively when installing, patching, upgrading, and managing Oracle systems. Table 1.21 shows several examples of how these variables define the locations of other Oracle directories.

TABLE 1.21 Common Uses of *ORACLE_BA*SE and *ORACLE_HOME*

Directory	Description
`$ORACLE_HOME/dbs`	Default location for PFILES and SPFILES on Unix systems
`%ORACLE_HOME%\database`	Default location for PFILES and SPFILES on Windows systems
`$ORACLE_BASE/admin/PROD/pfile`	Location of the PFILE for a database called `PROD` on Unix systems
`%ORACLE_BASE%\admin\PROD\pfile`	Location of the PFILE for a database called `PROD` on Windows systems
`$ORACLE_HOME/network/admin`	Default location for Oracle Net configuration files on Unix systems
`%ORACLE_HOME%\network\admin`	Default location for Oracle Net configuration files on Windows systems
`$ORACLE_HOME/rdbms/admin`	Location of many Oracle database configuration scripts on Unix systems
`%ORACLE_HOME%\rdbms\admin`	Location of many database configuration scripts on Windows systems

For Unix systems, Table 1.21 says `$ORACLE_HOME/dbs` is the default location for the PFILE and SPFILE, but then says that PFILES should be stored in `$ORACLE_BASE/admin/PROD/pfile`. Windows systems are similar. This implies that the same file needs to be in two locations at the same time. You can accomplish this using two tricks. Which you use depends on your operating system.

On Unix systems, you can create the PFILE in the `$ORACLE_BASE/admin/PROD/pfile` directory and then create a symbolic link in `$ORACLE_HOME/dbs` that points to the file in `$ORACLE_BASE/admin/PROD/pfile` using this syntax:

```
ln -s $ORACLE_BASE/admin/PROD/pfile/initPROD.ora
   $ORACLE_HOME/dbs/initPROD.ora
```

On Windows systems, you can create the PFILE in the `%ORACLE_BASE%\admin\PROD\pfile` directory and then put another PFILE in `%ORACLE_HOME%\database` that contains a single entry that points to the other PFILE in `%ORACLE_BASE%\admin\PROD\pfile` like this:

```
ifile=D:\oracle\admin\PROD\pfile\initPROD.ora
```

Using these techniques allows you to put the initialization parameter files in their default locations under `$ORACLE_HOME`, but also in their desired location under `$ORACLE_BASE`.

Other Administrative Directories

According to the OFA model, for a database called `PROD`, the initialization parameter file should be located in `$ORACLE_BASE/admin/PROD/pfile`. However, the OFA model also recommends that several other directories be built under this location for other management purposes. These directories are located under `$ORACLE_BASE/admin/PROD`:

adhoc This directory is designed to store ad hoc SQL scripts for the `PROD` database.

arch If the database is in archive log mode, this directory is specified as the location where the archived redo logs are to be written by LGWR. The PFILE/SPFILE parameter called `LOG_ARCHIVE_DEST` specifies the location of the `arch` directory.

adump If database auditing is turned on, this directory is specified as the location where the audit trail information is to be written. The PFILE/SPFILE parameter `AUDIT_FILE_DEST` specifies the location of the `adump` directory.

bdump This directory is the location of the database Alert log file and any trace files generated by background processes. The PFILE/SPFILE parameter `BACKGROUND_DUMP_DEST` specifies the location of the `bdump` directory.

cdump This directory is the location where core dump files will be written by operating system processes that crash. The PFILE/SPFILE parameter `CORE_DUMP_DEST` specifies the location of the `cdump` directory.

create This directory stores the SQL scripts that were used to initially create the database. These scripts may have been manually created or created using the Oracle Database Configuration Assistant described in Chapter 2.

exp This directory stores files that have been created using the Oracle `export` utility.

logbook This directory stores files that document the activities you performed on the database.

pfile This directory stores the parameter initialization file, or PFILE, for the database.

udump This directory is the location where any trace files generated by user processes will be written. The PFILE/SPFILE parameter `USER_DUMP_DEST` specifies the location of the `udump` directory.

NOTE

Why should the "real" copy of the PFILES be stored under $ORACLE_BASE instead of $ORACLE_HOME? Because it is a good idea to keep only version-specific files under $ORACLE_HOME. That way, when you eventually uninstall the software from an old $ORACLE_HOME, you won't lose your carefully tailored initialization files.

In addition to $ORACLE_BASE and $ORACLE_HOME, a few other non-OFA-related operating system environment variables on Unix and Windows systems are important to be aware of. These are described in Table 1.22.

TABLE 1.22 Common Non-OFA Environment Variables

Operating System Variable	Description
$ORACLE_SID	Defines which instance a Unix user session should be connecting to on the server.
%ORACLE_SID%	Defines which instance a Windows user session should connect to on the server.
$TNS_ADMIN	Specifies where the Oracle Net configuration files are stored on Unix systems—if they are to be stored outside their default location of $ORACLE_HOME\network\admin.
%TNS_ADMIN%	Specifies where the Oracle Net configuration files are stored on Windows systems—if they are to be stored outside their default location of %ORACLE_HOME%/network/admin.
$TWO_TASK	Establishes a default Oracle Net connection string that will be used on Unix systems if none is specified by the user.
%LOCAL%	Establishes a default Oracle Net connection string that will be used on Windows systems if none is specified by the user.
$LD_LIBRARY_PATH	Specifies the locations of the Oracle shared object libraries. This variable usually points to $ORACLE_HOME/lib or $ORACLE_HOME/lib32 on Unix systems.
$PATH	Tells the operating system in which directories to look for executable files on Unix systems.
%PATH%	Tells the operating system in which directories to look for executable files on Windows systems.

NOTE: The directories described in the "Other Administrative Directories" sidebar are merely guidelines. On large production databases, you will likely not want to store the database archived redo logs and export files under `$ORACLE_HOME/admin/PROD` because of their size, number, and need to be actively managed.

You'll need many of the environment settings discussed in this section in order to install Oracle 10*g*. You will see examples of their use later in this chapter, in the section "Responding to OUI Prompts."

Creating OFA Directories Structures

The OFA directory structure and its associated operating system environment variables are primarily used to manage the Oracle 10*g* software binaries and the supporting files for an instance such as the PFILE and SPFILES. However, the OFA model also prescribes a directory naming convention and structure for the physical database files. In general, the OFA model recommends that the physical database file be located in a directory structure like the one shown here for a database called `PROD` with five mount points:

- `/u01/oradata/PROD`
- `/u02/oradata/PROD`
- `/u03/oradata/PROD`
- `/u04/oradata/PROD`
- `/u05/oradata/PROD`

Notice that the directory naming convention incorporates the database name in the path. This makes it easier to have multiple files from multiple databases on the same mount points, on the same server. In addition to these directory paths, the names of the physical database files themselves also have an OFA naming convention, which is discussed in the following sections.

Control Files

Control files use names such as `control`*`n`*`.ctl`, in which *n* is the number of the copy of the multiplexed control file, for example, `control01.ctl`, `control02.ctl`, and so on.

NOTE: The query shown earlier on `V$CONTROLFILE` demonstrates an OFA-compliant naming convention for control files.

Datafiles

Datafiles use names such as `filename`*`n`*`.dbf`, in which *n* is the number of the datafile of a tablespace that is composed of multiple datafiles. The datafile names should also describe the tablespace to which they belong. For example, if a tablespace called `TOOLS` comprises four datafiles, those datafiles might be called `tools01.dbf`, `tools02.dbf`, `tools03.dbf`, and `tools04.dbf`.

NOTE Databases that use the Oracle Managed Files (OMF) feature will use system-generated filenames. See Chapter 3 for details.

NOTE The query shown earlier on `DBA_DATA_FILES` demonstrates an OFA-compliant naming convention for datafiles.

Redo Log Files

Redo logs use names such as `redogm.log`, in which *g* is the group number of the redo log and *m* is the member number. For example, if a database has three redo log groups, and each redo log group is multiplexed with two members, the first redo log group's files might be called `redo1a.log` and `redo1b.log`.

NOTE The query shown earlier on `V$LOGFILE` demonstrates an OFA-compliant naming convention for redo logs.

Many of these environment variables and OFA file locations are used by the Oracle Universal Installer during the installation process. An example of using the Oracle Universal Installer to install Oracle 10*g* is shown in the next section.

Using the Oracle Universal Installer

You use the *Oracle Universal Installer (OUI)* to install and configure the Oracle 10*g* software. The OUI is a Java-based application that provides the same installation look and feel no matter which operating system the install is being run on. The OUI process consists of six primary operations:

- Mounting the CD and starting the OUI
- Performing preinstallation checks
- Responding to server-specific prompts for file locations, names, and so on
- Selecting the products that you want to install
- Copying the files from the install media to `$ORACLE_HOME`
- Compiling the Oracle binaries
- Performing post-install operations using Configuration Assistants

Mounting the CD and Starting the OUI

To begin the install process, insert the Oracle 10*g* CD in the server. On some Unix systems, you may have to use the appropriate operating system command to mount the CD in your server before it is accessible.

Once the CD is mounted, take a moment to set the environment variables `$ORACLE_BASE`, `$ORACLE_HOME`, `$PATH`, and `$LD_LIBRARY_PATH`. OUI installations on Unix systems must also set the X Windows *`DISPLAY`* environment variable as shown here before continuing; otherwise, the OUI will not display correctly:

```
$ ORACLE_BASE=/u01/app/oracle; export ORACLE_BASE
$ ORACLE_HOME=$ORACLE_BASE/product/10.1.0; export ORACLE_HOME
$ PATH=$ORACLE_HOME/bin:$PATH; export PATH
$ LD_LIBRARY_PATH=$ORACLE_HOME/lib; export LD_LIBRARY_PATH
$ DISPLAY=192.168.1.100:0.0; export DISPLAY
```

NOTE After mounting the CD, you may want to copy its contents to a staging directory so that you can install from there instead of from the CD.

Performing Preinstallation Checks

Once these operating system variables are set, start the OUI using the `runInstaller.sh` command shown here:

```
$ /cdrom/runInstaller
Starting Oracle Universal Installer...

Checking installer requirements...

Checking operating system version: must be redhat-2.1, UnitedLinux-1.0
     or redhat-3
                                      Passed


All installer requirements met.

Preparing to launch Oracle Univeral Installer from
     /tmp/OraInstall2004-07-16_01-53-28PM. Please wait...
```

Notice that the output shows that the OUI checked the server's operating system version, available RAM, kernel settings, and so on that we described earlier in this chapter.

NOTE If needed, you can turn off the system verification that occurs prior to the installation by using the `-ignoreSysPrereqs` option of the `runInstaller` command.

Once the preinstallation tests are completed and passed, the OUI displays the initial OUI screen shown in Figure 1.20.

Click the Next button on the OUI screen to proceed with the installation.

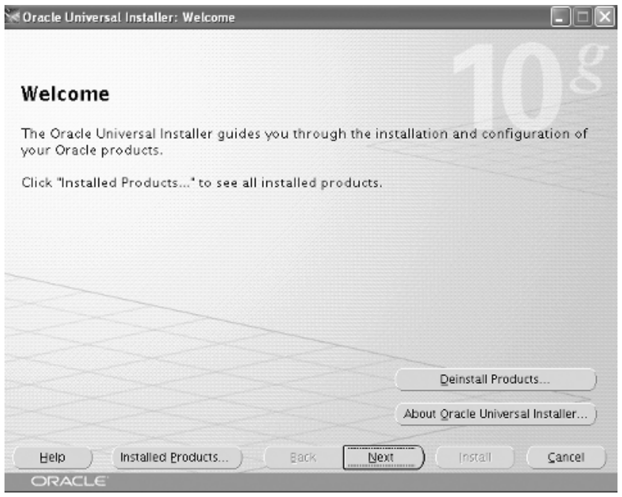
Responding to OUI Prompts

The next OUI screen prompts you for two pieces of information:

- The location for the inventory files that the OUI uses to keep track of which Oracle products are installed on the server
- The name of the operating system group of which the user doing the install is a member

You can see both items in Figure 1.21.

FIGURE 1.20 The initial OUI installation screen



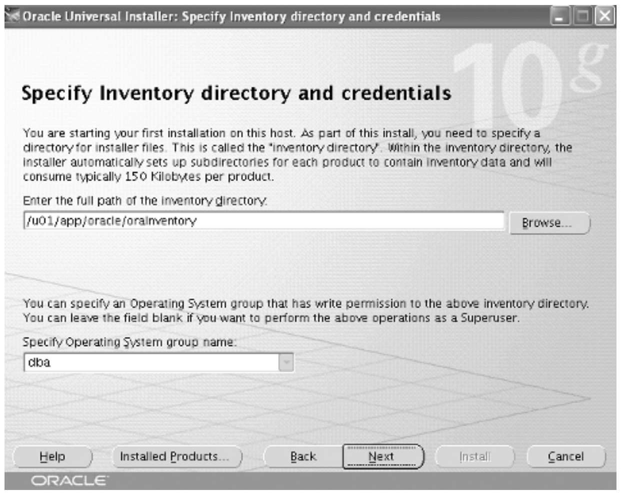
Oracle Universal Installer: Welcome

Welcome

The Oracle Universal Installer guides you through the installation and configuration of your Oracle products.

Click "Installed Products..." to see all installed products.

FIGURE 1.21 The OUI Specify Inventory Directory and Credentials screen



Oracle Universal Installer: Specify Inventory directory and credentials

Specify Inventory directory and credentials

You are starting your first installation on this host. As part of this install, you need to specify a directory for installer files. This is called the "inventory directory". Within the inventory directory, the installer automatically sets up subdirectories for each product to contain inventory data and will consume typically 150 Kilobytes per product.

Enter the full path of the inventory directory:
/u01/app/oracle/oraInventory

You can specify an Operating System group that has write permission to the above inventory directory. You can leave the field blank if you want to perform the above operations as a Superuser.

Specify Operating System group name:
dba

The value suggested for the oraInventory location, `/u01/app/oracle/oraInventory`, was selected based on the `$ORACLE_BASE` environment variable. The value suggested for the operating system group, `dba`, is the Oracle default value. Because both settings are correct for our environment, click the Next button to continue the installation. When you click the Next button on a Unix system, the OUI displays the screen shown in Figure 1.22.

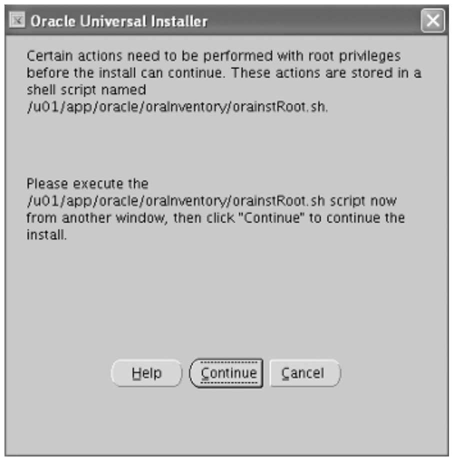
This screen asks you to open a second session on the Unix server as the superuser root so that the script `orainstRoot.sh` can be executed. The following example shows this `orainstRoot.sh` script being executed from another session:

```
$ su -
Password:
# cd /u01/app/oracle/oraInventory
./orainstRoot.sh
Creating the Oracle inventory pointer file (/etc/oraInst.loc)
Changing groupname of /u01/app/oracle/oraInventory to dba.
#
```

Running the script creates some directory structures that are used to support the Oracle installation and sets the proper file permissions on those directories as well as other files. Once the `orainstRoot.sh` script executes, click the Continue button to open the Specify File Locations screen, which is shown in Figure 1.23.

On this screen, you specify the location of the source CD and the destination location of the Oracle software (for example, `$ORACLE_HOME`). Additionally, you can assign a nickname to this `$ORACLE_HOME` (`OraDb10g_home1`, in this example). The value of `/u01/app/oracle/product/10.0.1` is filled in automatically by the installer based on the `$ORACLE_HOME` environment variable setting that you established before starting the OUI. Click the Next button to open the Select Installation Type screen, as shown in Figure 1.24, and continue with the installation.

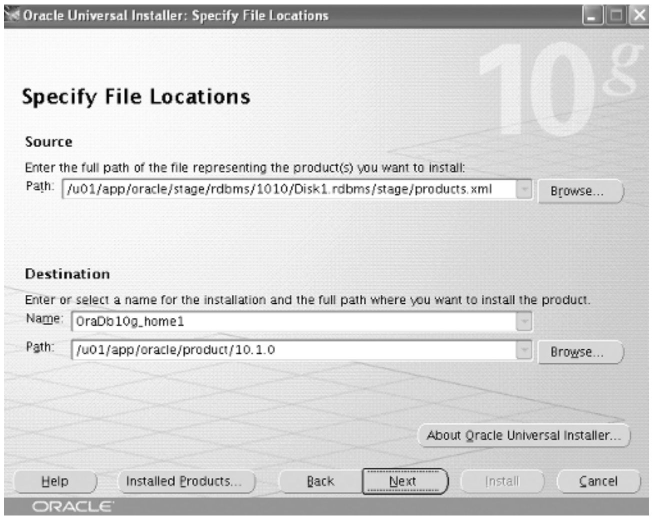
FIGURE 1.22 The OUI *orainstRoot.sh* screen



Certain actions need to be performed with root privileges before the install can continue. These actions are stored in a shell script named /u01/app/oracle/oraInventory/orainstRoot.sh.

Please execute the /u01/app/oracle/oraInventory/orainstRoot.sh script now from another window, then click "Continue" to continue the install.

FIGURE 1.23 The Specify File Locations screen


Oracle Universal Installer: Specify File Locations

Specify File Locations

Source

Enter the full path of the file representing the product(s) you want to install:

Path: /u01/app/oracle/stage/rdbms/1010/Disk1.rdbms/stage/products.xml

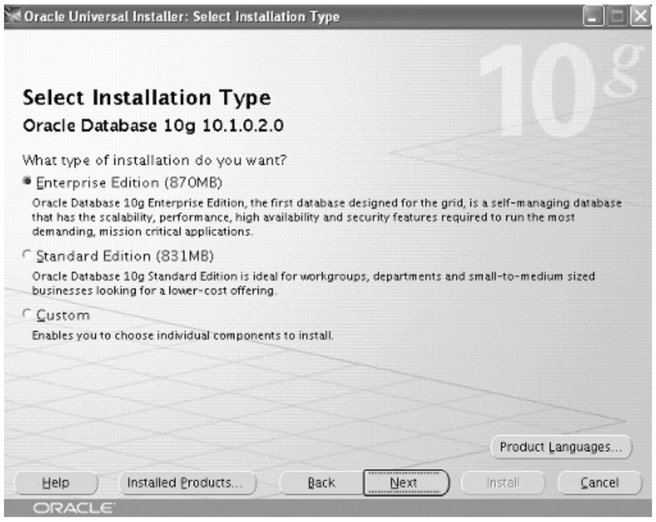
Destination

Enter or select a name for the installation and the full path where you want to install the product.

Name: OraDb10g_home1

Path: /u01/app/oracle/product/10.1.0

FIGURE 1.24 The Select Installation Type screen


Oracle Universal Installer: Select Installation Type

Select Installation Type

Oracle Database 10g 10.1.0.2.0

What type of installation do you want?

[✓ Enterprise Edition (870MB)]

Oracle Database 10g Enterprise Edition, the first database designed for the grid, is a self-managing database that has the scalability, performance, high availability and security features required to run the most demanding, mission critical applications.

Standard Edition (831MB)

Oracle Database 10g Standard Edition is ideal for workgroups, departments and small-to-medium sized businesses looking for a lower-cost offering.

Custom

Enables you to choose individual components to install.

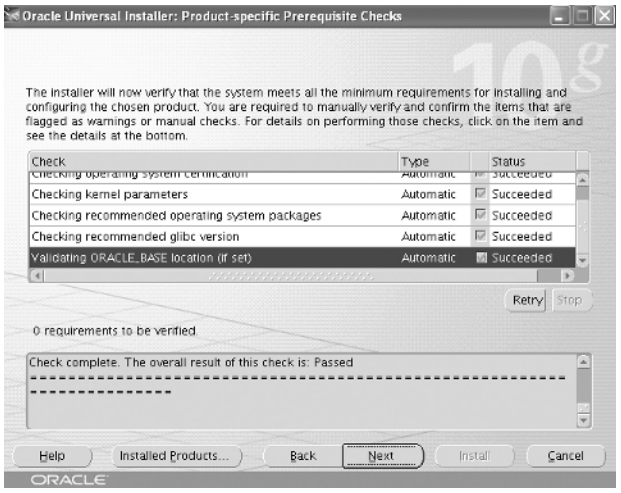
Selecting Products to Install

This screen prompts you to select the type of installation to perform. In this example, we selected the Enterprise Edition option.

Click the Next button to open the next screen, which is shown in Figure 1.25.

The OUI goes thorough a second round of installation checks that confirm that the server's operating system version and configuration are appropriate for the Enterprise Edition install of Oracle 10*g*. If all the verification checks complete successfully, click the Next button to open the Select Database Configuration screen, as shown in Figure 1.26.

FIGURE 1.25 The OUI verifies system version and configuration.


Oracle Universal Installer: Product-specific Prerequisite Checks

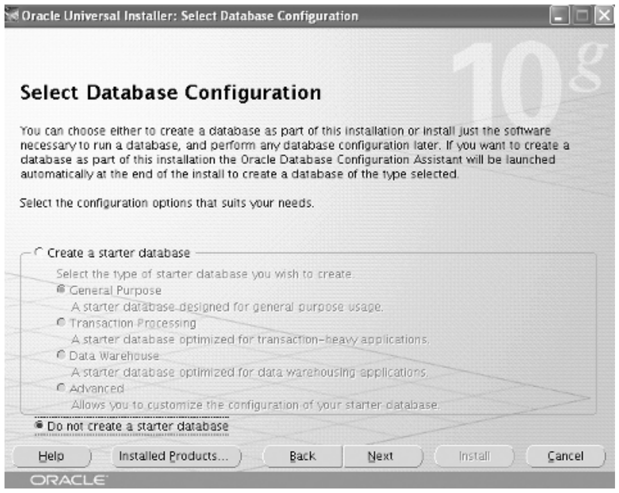
The installer will now verify that the system meets all the minimum requirements for installing and configuring the chosen product. You are required to manually verify and confirm the items that are flagged as warnings or manual checks. For details on performing those checks, click on the item and see the details at the bottom.

Check	Type		Status
Checking operating system certification	Automatic	✓	Succeeded
Checking kernel parameters	Automatic	✓	Succeeded
Checking recommended operating system packages	Automatic	✓	Succeeded
Checking recommended glibc version	Automatic	✓	Succeeded
Validating ORACLE_BASE location (if set)	Automatic	✓	Succeeded

0 requirements to be verified.

```
Check complete. The overall result of this check is: Passed
============================================================
===============
```

FIGURE 1.26 The Select Database Configuration screen


Oracle Universal Installer: Select Database Configuration

Select Database Configuration

You can choose either to create a database as part of this installation or install just the software necessary to run a database, and perform any database configuration later. If you want to create a database as part of this installation the Oracle Database Configuration Assistant will be launched automatically at the end of the install to create a database of the type selected.

Select the configuration options that suits your needs.

- ○ Create a starter database
  - Select the type of starter database you wish to create.
  - ● General Purpose — A starter database designed for general purpose usage.
  - ○ Transaction Processing — A starter database optimized for transaction-heavy applications.
  - ○ Data Warehouse — A starter database optimized for data warehousing applications.
  - ○ Advanced — Allows you to customize the configuration of your starter database.
- ● Do not create a starter database

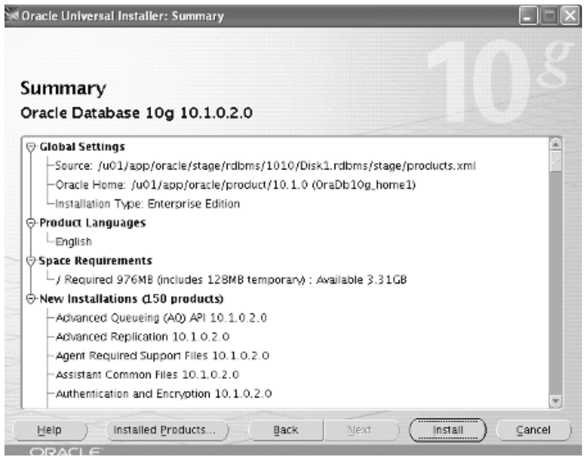
This screen asks if you want to create a database following the installation process. Because creating a database is covered in Chapter 2, we'll skip this step for now. Click the Do Not Create A Starter Database option, and then click Next to open the Summary screen, as shown in Figure 1.27.

This screen summarizes all the options that you selected and all the components that will be installed. If you need to make changes, click the Back button to go back and modify your previous selections. If you are satisified with your selections, click the Next button to start copying the Oracle binaries from the CD to the `$ORACLE_HOME` directory.

Copying and Compiling Files

As shown in Figure 1.28, a bar is displayed to indicate the progress of the copy process.

FIGURE 1.27 The Summary screen

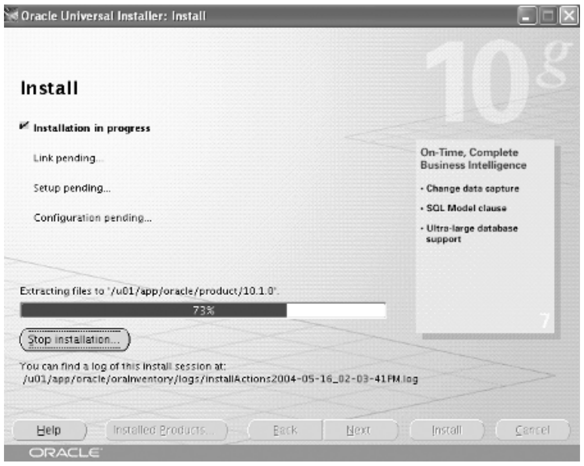


Summary

Oracle Database 10g 10.1.0.2.0

```
Global Settings
  Source: /u01/app/oracle/stage/rdbms/1010/Disk1.rdbms/stage/products.xml
  Oracle Home: /u01/app/oracle/product/10.1.0 (OraDb10g_home1)
  Installation Type: Enterprise Edition
Product Languages
  English
Space Requirements
  / Required 976MB (includes 128MB temporary) : Available 3.31GB
New Installations (150 products)
  Advanced Queueing (AQ) API 10.1.0.2.0
  Advanced Replication 10.1.0.2.0
  Agent Required Support Files 10.1.0.2.0
  Assistant Common Files 10.1.0.2.0
  Authentication and Encryption 10.1.0.2.0
```

FIGURE 1.28 To track the copy progress, keep an eye on the bar.



Install

✓ Installation in progress

Link pending...

Setup pending...

Configuration pending...

On-Time, Complete Business Intelligence
- Change data capture
- SQL Model clause
- Ultra-large database support

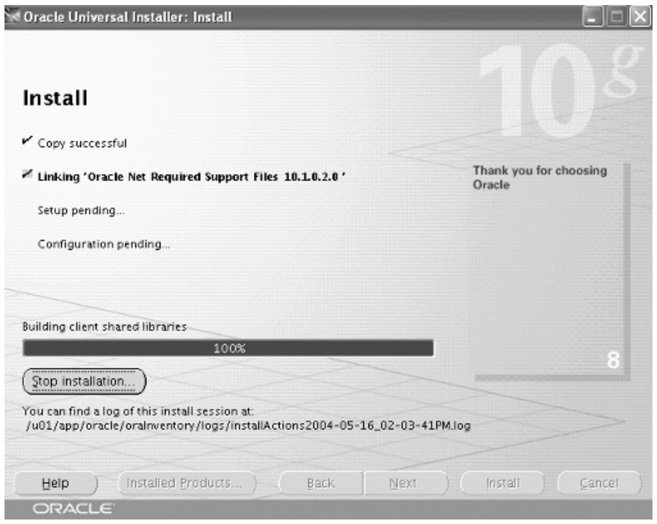
Extracting files to '/u01/app/oracle/product/10.1.0'.

You can find a log of this install session at:
/u01/app/oracle/oraInventory/logs/installActions2004-05-16_02-03-41PM.log

NOTE If these operating system checks do not succeed, you must correct the areas failing the checks before continuing.

Once the file copy portion of the installation is complete, the OUI begins linking the binaries to create the executable files needed to make the Oracle 10*g* software run on the server. Figure 1.29 shows the linking progress bar.

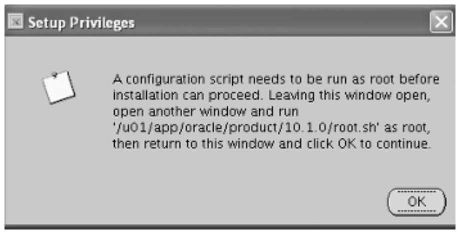
FIGURE 1.29 You can track the linking process by watching the bar.


Oracle Universal Installer: Install
Install
✓ Copy successful
✓ Linking 'Oracle Net Required Support Files 10.1.0.2.0'
Setup pending...
Configuration pending...
Thank you for choosing Oracle
Building client shared libraries
100%
Stop installation...
Help
Installed Products...
Back
Next
Install
Cancel
ORACLE

You can find a log of this install session at:
/u01/app/oracle/oraInventory/logs/installActions2004-05-16_02-03-41PM.log

On Unix systems, following the linking process, you are prompted to execute a second configuration script as the superuser root from the Unix command line, as shown in Figure 1.30.

FIGURE 1.30 Running the script as root



A configuration script needs to be run as root before installation can proceed. Leaving this window open, open another window and run '/u01/app/oracle/product/10.1.0/root.sh' as root, then return to this window and click OK to continue.

Like the first root script, the `root.sh` script should be executed from a second session as shown here:

```
$ su -
Password:
```

```
# cd /u01/app/oracle/product/10.1.0
./root.sh
Running the Oracle10 root.sh script...
The following environment variables are set as:
  ORACLE_OWNER= oracle
  ORACLE_HOME= /u01/app/oracle/product/10.1.0

Enter the full pathname of the local bin directory: [/usr/local/bin]:

Creating /etc/oratab file...
Adding entry to /etc/oratab file...
Entries will be added to the /etc/oratab file as needed by
Database Configuration Assistant when a database is created
Finished running generic part of root.sh script.
Now product-specific root actions will be performed.
Successfuly accumulated necessary OCR keys.
Creating OCR keys for the user 'root', privgrp 'root'..
Operation successful.
Oracle Cluster Registry for cluster has been initialized

Adding to inittab
Checking the status of Oracle init process...
Expecting the CRS daemons to be up within 600 seconds.
CSS is active on these nodes.
        moderne
CSS is active on all nodes.
Oracle CSS service is installed and running under init(1M)
#
```

Executing the `root.sh` script copies some files to a location outside `$ORACLE_HOME` and sets the permissions on several files inside and outside `$ORACLE_HOME`. Once the `root.sh` script executes successfully, click OK to continue the installation.

NOTE

If you have multiple installations to perform, you can speed up the process and minimize errors by building an OUI response file. This text file contains all the necessary responses to the OUI prompts so that an unattended, silent install is possible.

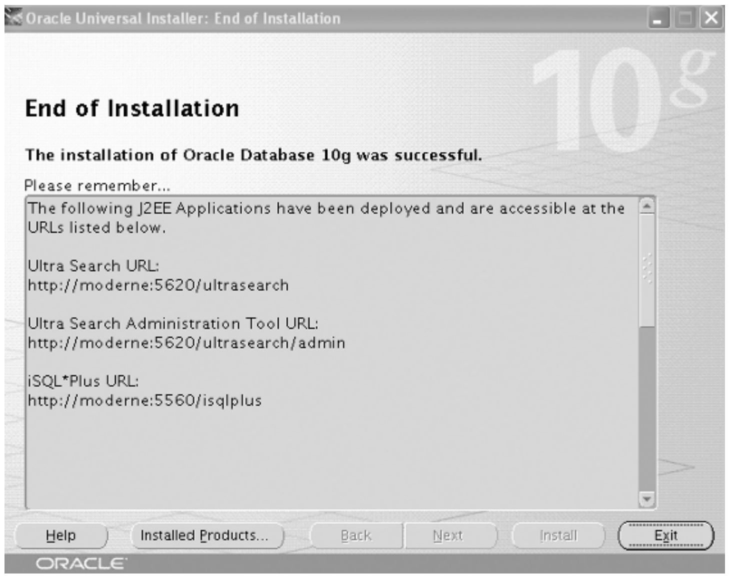
NOTE

The OUI on Windows systems also offers a Basic Installation mode in which only a few installation questions are asked before the copying of files begins. If you select the Advanced Installation mode, the prompts will closely follow those shown for Unix in this section.

Performing Post-Install Tasks

Once the `root.sh` script has completed, the OUI will perform some brief post-installation configuration activities before displaying the End Of Installation screen shown in Figure 1.31.

FIGURE 1.31 The End Of Installation Screen



End of Installation

The installation of Oracle Database 10g was successful.

Please remember...

```
The following J2EE Applications have been deployed and are accessible at the
URLs listed below.

Ultra Search URL:
http://moderne:5620/ultrasearch

Ultra Search Administration Tool URL:
http://moderne:5620/ultrasearch/admin

iSQL*Plus URL:
http://moderne:5560/isqlplus
```

Click the Exit button and then the OK button on the pop-up screen to exit the OUI and return to the Unix prompt.

Once the OUI is complete, you should have a completely installed and configured `$ORACLE_HOME`. We'll use this software to create your first database in Chapter 2.

Summary

Oracle offers a broad product mix that includes not only their popular database software, but also application servers, development tools, and enterprise-level applications and services. At the heart of Oracle's product mix is the relational database model that uses tables to store application data and constraints to enforce relationships between tables and implement business rules. Examples of these constraints include Primary and Foreign Key, Not Null and Unique, and Check.

You use SQL to retrieve, add, modify, and delete data in Oracle tables. SQL statements are broadly categorized as query, DML, DDL, or DCL commands. `SELECT` statements are composed of the `SELECT` and `FROM` clauses and, optionally, `WHERE`, `HAVING`, and `ORDER BY` clauses. `INSERT`, `UPDATE`, and `DELETE` commands are examples of SQL DML commands. You can enter these commands directly through tools such as SQL*Plus or *i*SQL*Plus or through applications such as Enterprise Manager and compiled C++, COBOL, or Java programs. `CREATE`, `ALTER`, and `DROP` are examples of SQL DDL commands, and `GRANT` and `REVOKE` are examples of SQL DCL commands. Oracle also provides a procedural extension to SQL called PL/SQL.

Oracle uses a number of data dictionary views and dynamic performance views to provide information about the database and its contents. These contents include segments of many types, including tables, indexes, and partitions. A segment is composed of extents, which are composed of database blocks, which are composed of operating system blocks.

Extents are stored within tablespaces. A database must have at least three tablespaces, `SYSTEM`, `SYSAUX`, and `TEMP`, but may have many more depending on the application that it supports. Each tablespace is associated with a physical file or files on disk called a datafile. The datafiles store the data that has been inserted into application tables. Other files that make up the Oracle database include control files and redo log files.

The datafiles are accessed by the user's Server Process whenever a SQL command is issued against the database. The user's Server Process parses each SQL statement and places it into the Shared Pool component of the SGA before copying the database blocks that contain the desired data into the Database Buffer Cache component of the SGA. If the user performs a DML statement, the recovery information needed to reproduce this statement is placed in the Redo Log Buffer component of the SGA where it is ultimately written to disk by the LGWR background process. Other processes that are part of the Oracle instance include Database Writer, System Monitor, Process Monitor, and Checkpoint. The parameter initialization file (PFILE or SPFILE) is used to configure the instance.

The OFA model is useful for establishing a manageable directory structure for a new Oracle server. The OFA model recommends mount point, directory, and file-naming conventions. Once the OFA structure is established, you can use the OUI to install the Oracle 10*g* software into the location you've selected.

Exam Essentials

Describe the Oracle tools and what they are used for. Know which tools are available for connecting to and interacting with an Oracle database. Understand how these tools differ from one another.

Understand relational database concepts. Understand important relational database concepts such as primary and foreign keys and how these are used to enforce business rules within the database.

Identify how SQL is used to interact with the database. Be able to describe how SQL and *i*SQL*Plus, GUI management tools, Java, PL/SQL, and Oracle's programming interfaces can be used to connect to an Oracle database; describe the structure of the tables; and use `SELECT`, `INSERT`, `UPDATE`, and `DELETE` statements to perform transactions.

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.

Explain Oracle 10*g* system requirements. Know what the requirements are for available server disk space and memory prior to performing an Oracle 10*g* installation.

Describe the Optimal Flexible Architecture. Be able to explain the concepts associated with the OFA model and how to implement an OFA-compliant installation and database directory structure.

Identify common environment variables. Know which environment variables are generally required for a successful Oracle 10*g* installation and what each variable is used for.

Describe steps for installation and configuration. Know how to set up the Oracle installation environment so that the OUI can be used to install and configure the Oracle 10*g* software.

Review Questions

1. Which of the following SQL commands is an example of a DML command?

A. `SELECT`

B. `CREATE`

C. `INSERT`

D. `GRANT`

2. You have just started a database transaction by inserting a row into a table. Which of the following actions will end this transaction?

A. Inserting another row

B. Issuing a `COMMIT` command

C. Issuing an `END TRANSACTION` command

D. Deleting the row you just inserted

3. You are designing an application that will enforce business rules through table design. One of the tables in the application contains information about parts that are used to manufacture your product. When creating the `PARTS` table, what could you do to make certain that each part receives a part number and that each part number will be unique?

A. Place a Unique constraint on the part number column of the `PARTS` table.

B. Place a Not Null constraint on the part number column of the `PARTS` table.

C. Place a Primary Key constraint on the part number column of the `PARTS` table.

D. Place a Foreign Key constraint on the part number field of the `PARTS` table.

4. Which of the following is not an advantage of SQL over traditional programming languages?

A. SQL statements use English-like commands.

B. A user can choose from several interfaces when interacting with SQL.

C. SQL has sophisticated condition testing and looping capabilities.

D. Users do not have to know how or where their data is physically stored in order to access it.

5. How much free disk space is required to install Oracle Database 10*g*?

A. 1.5MB

B. 15GB

C. 1.5KB

D. 1.5GB

6. You've just been hired as a DBA for a large company. During the interview process, you were shown the job description for the position. Which of the following tasks might have been included on this job description?

A. Install and configure Oracle 10*g* software.

B. Implement database installations according to OFA guidelines.

C. Use OFA-compliant naming conventions for database files and directories.

D. Any of the above may have been included on the DBA job description.

7. Which of the following is not one of the differences between SQL*Plus and *i*SQL*Plus?

A. *i*SQL*Plus is accessed via a web browser, but SQL*Plus must be run from a client or on the server.

B. *i*SQL*Plus output is displayed in a separate window from the commands, but SQL*Plus displays everything in one window.

C. *i*SQL*Plus can use SQL and PL/SQL to access the database, as can SQL*Plus.

D. *i*SQL*Plus can only be used to access databases on Windows servers, but SQL*Plus can be used to access databases on Unix or Windows servers.

8. Which of the following is an example of an environment variable that might be defined on a Unix system prior to starting the OUI?

A. `ORACLE_DIR`

B. `$ORACLE_HOME`

C. `$ORA_HOME`

D. `$ORACLE_HM`

9. The `DBA_TABLES` data dictionary view contains a column called `OWNER` that shows the user name of the account who owns each table. What SQL clause would you add to a query on `DBA_TABLES` to view only those tables owned by a user called `APPS`?

A. `WHERE`

B. `GROUP BY`

C. `HAVING`

D. `ALTER`

10. You have just logged on to the database as the user `APPS` and need to know which tables you own or have been granted access to. Which of the following data dictionary views would you query to see a listing of these tables?

A. `DBA_TABLES`

B. `USER_TABLES`

C. `ALL_TABLES`

D. `MY_TABLES`

11. You want to configure the instance so that the sizes of the Shared Pool, Buffer Cache, Large Pool, and Java Pool are automatically managed by Oracle. Which initialization parameter do you need to set to allow this to happen?

A. `SGA_TARGET`

B. `SHARED_POOL_SIZE`

C. `LARGE_POOL_SIZE`

D. `DB_CACHE_SIZE`

12. Which of the following is not considered part of an Oracle database?

A. Datafiles

B. Redo logs

C. PFILE and SPFILE

D. Control files

13. Which of the following statements is not always true?

A. Every database has at least three tablespaces.

B. Every database has at least three datafiles.

C. Every database has at least three multiplexed redo logs.

D. At least three types of segments can exist in a database.

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 datafile 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 is made up of memory structures and processes, whereas a database is composed of physical files.

B. An instance is only used 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 an 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. You've been asked to install Oracle 10*g* on a new Unix server. You're likely to ask the Unix system administrator to do all but which of the following for you in order to get the new server ready for Oracle?

A. Modify the server's kernel parameters.

B. Create a new Unix user to own the Oracle software.

C. Create the mount points and directory structure using the OFA model.

D. Determine which directory will be used for `$ORACLE_HOME`.

18. Oracle's OFA model specifies a naming convention for all but which of the following?

A. User names

B. Mount points

C. Directory paths

D. Database filenames

19. The Oracle Universal Installer is started by executing which program?

A. `emctl`

B. `runInstaller`

C. `ouistart`

D. `isqlplusctl`

20. On Unix systems, the script `root.sh` must be executed during the installation process. What is the purpose of this script?

A. It creates the root user in the database.

B. It creates the root directory for the server.

C. It grants root superuser privileges to the Oracle Unix account.

D. It copies files and sets permissions on files outside `$ORACLE_HOME`.

Answers to Review Questions

1. C. Examples of DML, or data manipulation commands, include `INSERT`, `UPDATE`, and `DELETE`. DDL, or data definition language commands, include `CREATE`, `ALTER`, or `DROP`. DCL, or data control language commands, include `GRANT` and `REVOKE`.

2. B. Transactions are ended when the transaction is either committed using the `COMMIT` command, or rolled back using the `ROLLBACK` command.

3. C. Tables with a Primary Key constraint on them require that any rows inserted into the table have a value that is unique and not null for the primary key column.

4. C. SQL does not provide any condition testing or looping capabilities. Oracle provides PL/SQL to allow for these types of programming constructs.

5. D. Oracle 10*g* requires 1.5GB of free disk space for installation. An additional 1GB is required if a sample database is created using the OUI.

6. D. The tasks that a DBA performs encompass all these areas plus managing database storage, security, and availability.

7. D. *i*SQL*Plus can be used to access databases on both Unix and Windows servers, as can SQL*Plus.

8. B. The `$ORACLE_HOME` environment variable would point to the location where the Oracle binaries should be installed.

9. A. The SQL `WHERE` clause qualifies the results of the query and limits the rows that are returned to those that match the condition specified in the `WHERE` clause.

10. C. The `ALL_TABLES` view displays information about the tables that the user owns or has been granted access to. The `DBA_TABLES` view displays information about all tables in the database. The `USER_TABLES` view shows only information about the user's own tables. There is no view called `MY_TABLES`.

11. A. Setting `SGA_TARGET` to a nonzero value in the PFILE or SPFILE allows the automated memory management feature to dynamically assign space to these four areas of the SGA.

12. C. Although PFILEs and SPFILEs are physical files used to configure the Oracle instance, they are not considered part of the database.

13. C. Every database must have at least two redo log files, which may or may not be multiplexed.

14. D. The LRU mechanism ensures that each user's Server Process can find free space in the Shared Pool and Database Buffer Cache whenever they need it, but also keeps frequently used objects cached in those memory areas.

15. A. The instance consists of the SGA and all the Oracle background processes. The database is composed of the control files, datafiles, and redo logs.

16. B. Multiple operating system blocks make up database blocks, contiguous chunks of which make up extents, which comprise segments.

17. D. While the Unix system administrator is responsible for creating volume groups and mount points, the DBA generally decides where the Oracle binaries will be installed—the location designated by the `$ORACLE_HOME` environment variable.

18. A. The OFA model does not include any reference to naming conventions for things inside the database, such as users, tables, or tablespaces.

19. B. The `runInstaller` executable performs a preinstall check of the operating system and hardware resources before starting the OUI graphical tool.

20. D. The `root.sh` script copies configuration files to directories outside `$ORACLE_HOME` and sets the permissions on those files accordingly.