Database Programming

Creating Tables

Objectives

In this lesson, you will learn to:

- List and categorize the main database objects
- Review a table structure
- Describe how schema objects are used by the Oracle database
- List and provide an example of each of the number, date, and character data types

Objectives (cont.)

In this lesson, you will learn to:

- Create a table using the appropriate data type for each column
- Explain the use of external tables
- Query the Data Dictionary to obtain the names and other attributes of database objects

Purpose

Up until now, you have selected, updated, inserted, and deleted information in the existing tables of a database. As a Database Administrator (DBA), you will be expected to know how to create tables as well.

Purpose (cont.)

In this lesson, you will learn which database objects are most frequently used, how to look at the table structure, and how to create new tables. Your tables will be small compared to tables that hold millions of rows and hundreds of columns, but creating a small table involves the same SQL statements and syntax as creating a very large one.

You will also learn about external tables—tables which are similar in structure to the normal Oracle database tables, but the actual data rows are stored externally in a flat file and are accessed only when needed.

Database Schema Objects

An Oracle Database can contain many different types of objects. This section introduces the most commonly used objects, and also describes how the Oracle Server uses the information stored in the Data Dictionary when it is performing work as a result of the SQL statements you issue.

Database Schema Objects (cont.)

The main database object types are:

- Table
- Index
- Constraint
- View
- Sequence
- Synonym

Some of these object types can exist independently and others can not.

Database Schema Objects (cont.)

Some of the object types take up space, known as storage, in the database and others do not. Database objects taking up significant storage space are known as Segments. Tables and Indexes are examples of Segments, as the values stored in the columns of each row take up significant physical disk space.

Views, Constraints, Sequences, and Synonyms are also objects, but the only space they require in the database is in the definition of the object—none of them have any data rows associated with them.

Database Schema Objects (cont.)

The database stores the definitions of all database objects in the Data Dictionary, and these definitions are accessible to all users of the database as well as to the database itself.

Have you ever wondered how Oracle knows which columns to return from a Query? For example, if you specify SELECT * FROM d_cds instead of SELECT cd_number, title FROM d_cds, how does Oracle know which columns to return?

Database Schema Objects (cont.)

The database looks up the definition of the table used in the query, translates the ‘*’ into the full list of columns, and returns the result to you.

Database Schema Objects (cont.)

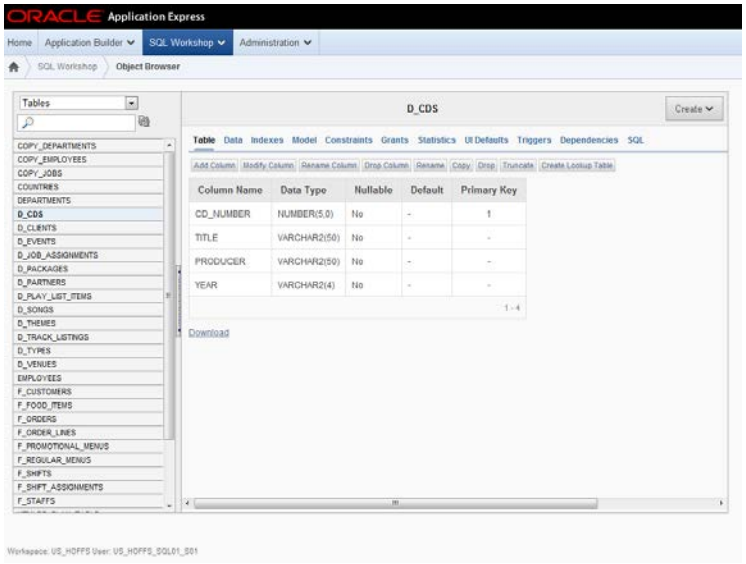
The database uses the Data Dictionary for all statements you issue, even if you list the column names instead of using '*'. It checks that the tables you are referencing in your statement exist in the database, it checks that the column names are correct, it checks if you have the correct privileges to perform the action you are requesting, and finally it uses the Data Dictionary to decide the Execution Plan – how it will actually perform the request.

Database Schema Objects (cont.)

The Data Dictionary itself can be queried by all database users. In Application Express, it can be accessed both via SQL statements in the SQL Workshop> SQL Commands interface and also from the SQL Workshop> Object Browser interface.

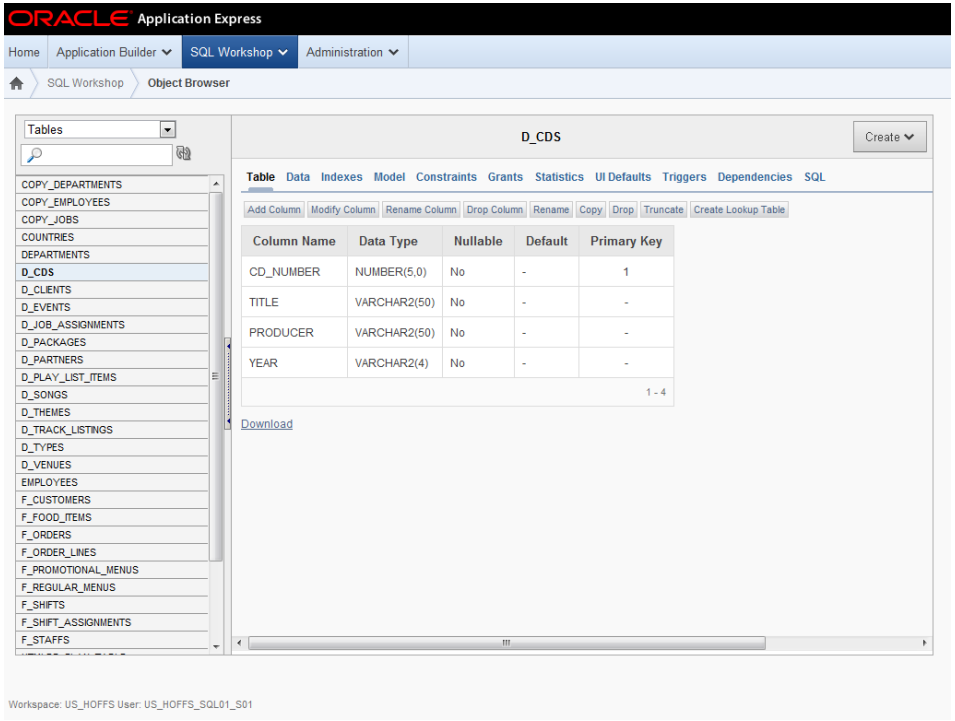
Database Schema Objects (cont.)

In the SQL Commands window, you have to know the names of the views you are querying, and in the Object Browser interface, you simply click on the listed objects to see their details. So if you want to see the details of the D_CDS table, you simply click on its title in the table listing:



Database Schema Objects (cont.)

In this example using the Object Browser, you can see the details of the D_CDS table as well as the options to view the data, indexes, constraints, grants, and other details of the table.



Database Schema Objects (cont.)

Using the SQL Commands window, you must ask for a DESCription of the table. All the extra options offered by Object Browser are not available in this interface.

Table Creation

All data in a relational database is stored in tables. When creating a new table, use the following rules for table names and column names:

- Must begin with a letter
- Must be 1 to 30 characters long
- Must contain only A - Z, a - z, 0 - 9, _ (underscore), $, and #
- Must not duplicate the name of another object owned by the same user
- Must not be an Oracle Server reserved word

Naming Conventions

It is best to use descriptive names for tables and other database objects. If a table will store information about students, name it STUDENTS, not PEOPLE or CHILDREN.

Also, names are not case sensitive. For example, STUDENTS is treated the same as STuDents or students.

Naming Conventions (cont.)

Creating tables is part of SQL's data definition language (DDL). Other DDL statements used to set up, change, and remove data structures from tables include ALTER, DROP, RENAME, and TRUNCATE.

CREATE TABLE

To create a new table, you must have the CREATE TABLE privilege and a storage area for it. The database administrator uses data control language (DCL) statements to grant this privilege to users and assign a storage area.

CREATE TABLE (cont.)

Tables belonging to other users are not in your schema. If you want to use a table that is not in your schema, use the table owner's name as a prefix to the table name:

```
SELECT *
FROM mary.students;
```

You must be granted access to a table to be able to select from it.

CREATE TABLE Syntax

To create a new table, use the following syntax details:

- **table** is the name of the table
- **column** is the name of the column
- **datatype** is the column's data type and length
- **DEFAULT expression** specifies a default value if a value is omitted in the INSERT statement

```
CREATE TABLE table
(column datatype [DEFAULT expression],
(column datatype [DEFAULT expression],
(……[ ] );
```

CREATE TABLE Example

The example below shows a CREATE TABLE statement:

```
CREATE TABLE cd_collection
(cd_number NUMBER(2),
   title VARCHAR2(14),
   artist VARCHAR2(13),
   purchase_date DATE DEFAULT SYSDATE);
```

Creating a Table Using a Subquery

A second method for creating a table is to apply the AS subquery clause, which both creates the table and inserts rows returned from the subquery.

This is an easy way to create a copy of a table to practice SQL statements. Note that you need to create a column alias for those columns in the subquery that contain expressions.

Creating a Table Using a Subquery (cont.)

Only datatype definitions and NOT NULL constraints are passed on to a new table created from a subquery. This is because the new table could be used in a different context, in which existing PK-FK relationships may not be relevant.

Creating a Table Using a Subquery Syntax

Syntax:

```
CREATE TABLE tablename
  [(column, column, …)]
   AS subquery;
```

Note: The column list is optional unless you want the column names to differ from the subquery column names.

Creating a Table Using a Subquery Examples

```
CREATE TABLE copy_mytable
AS
(SELECT code, name, start_date, end_date, give_away
FROM  f_promotional_menus);
```

```
CREATE TABLE dept80
AS
SELECT employee_id, last_name, salary*12 ANNSAL, hire_date
FROM  employees
WHERE department_id = 80;
```

Creating a Table Using a Subquery (cont.)

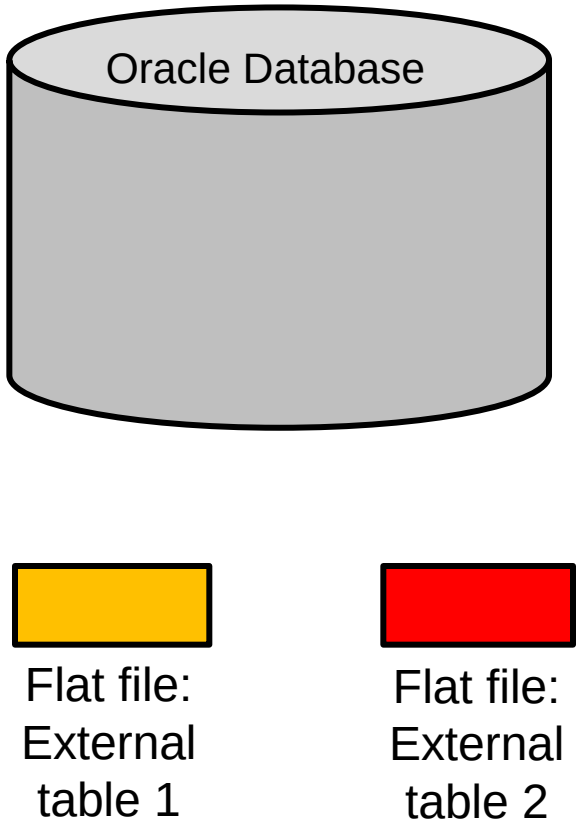
When a copy of a table is made using a subquery, the following rules are important:

- The column names in the new table will be identical to those in the original table, unless column aliases are used
- The column datatypes in the new table will be identical to those in the original table
- Only Not Null constraints are copied to the table; no other constraint types will exist on the new table

External Tables

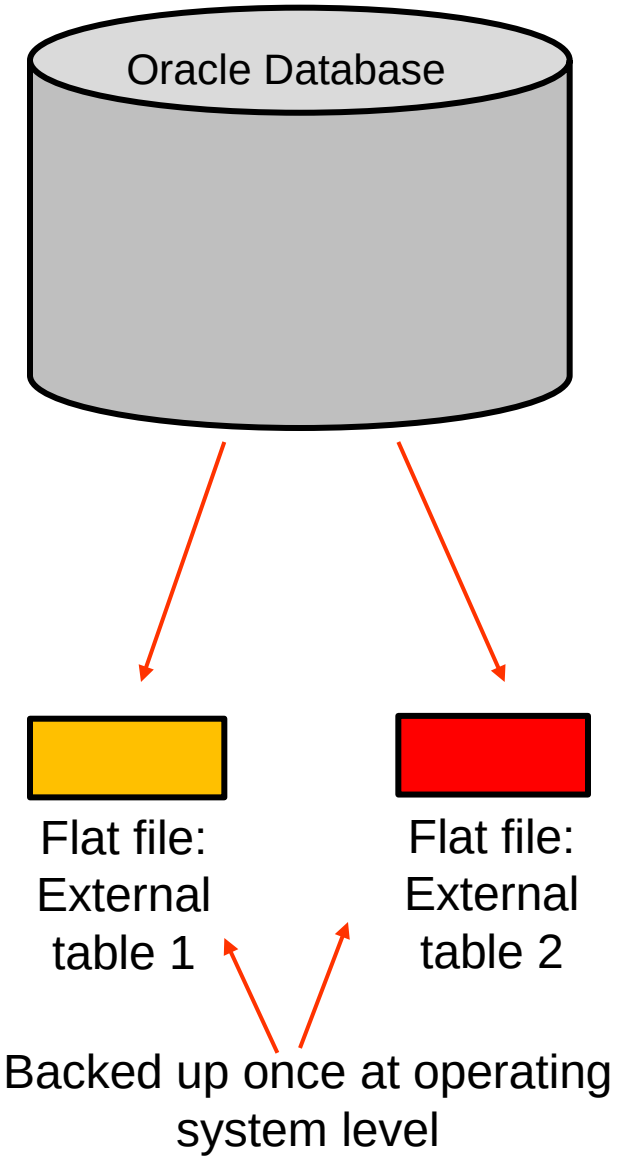
Oracle also supports another table type: External table.

In an external table, the data rows are not held inside the database files but are instead found in a flat file, stored externally to the database.



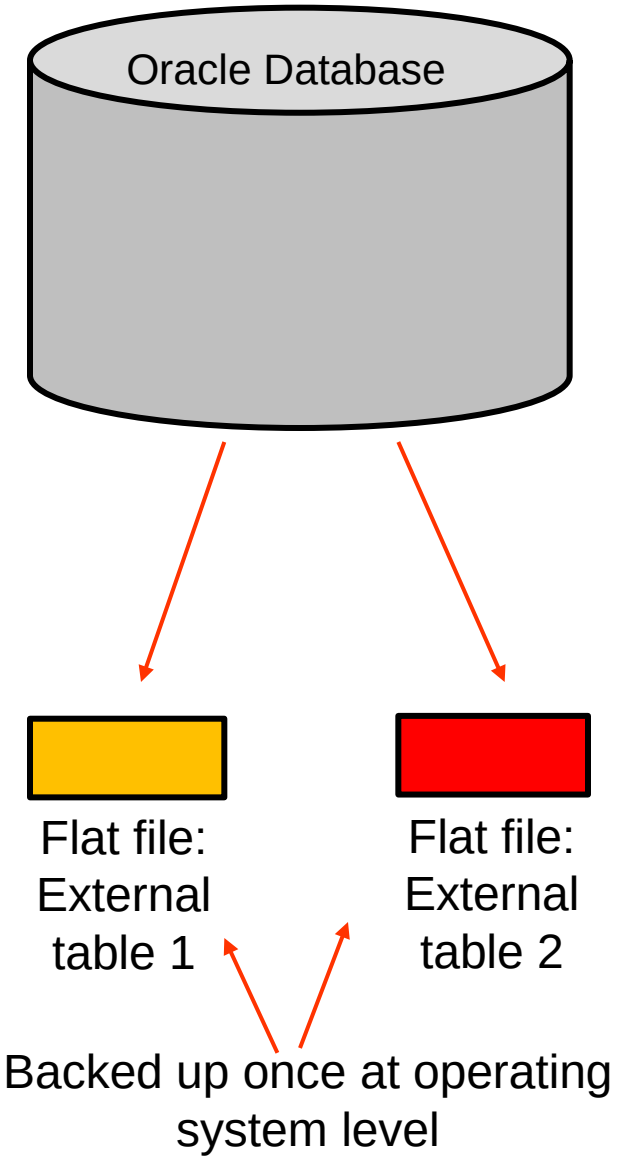
External Tables (cont.)

Typically an external table is used to store data migrated from older versions of the databases used by a company.



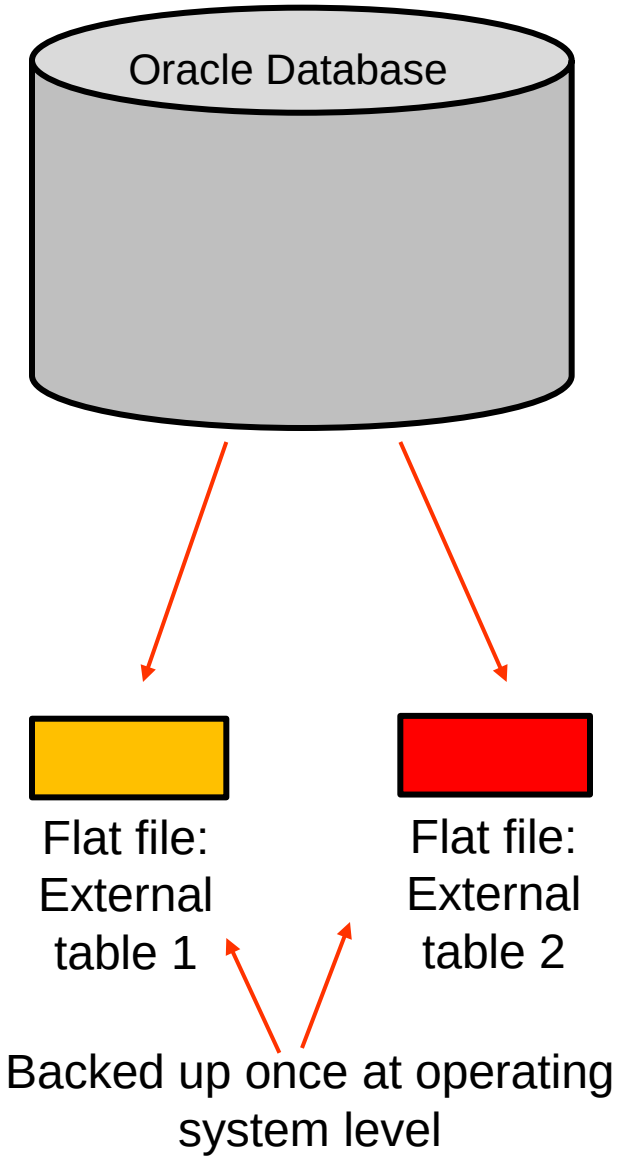
External Tables (cont.)

When a company is implementing a new application and database, they typically need to import most of the data from the old systems to the new system for normal read and write access, but there may be some data that is not used frequently and therefore would only need to be accessed for read access. This kind of data could be held in an external table.



External Tables (cont.)

One of the many benefits for Oracle is that data held in external tables only has to be backed up once, and then never again unless the contents of the file change.



External Tables (cont.)

The syntax to create an external table is very similar to that of creating a standard table, except that it has extra syntax at the end. Please note the new syntax, not used in standard SQL statements for table creation.

External Tables (cont.)

ORGANIZATION EXTERNAL -- tells Oracle to create an external table

TYPE ORACLE_LOADER -- of type Oracle Loader (an Oracle Product)

DEFAULT DIRECTORY def_dir1 -- the name of the directory for the file

External Tables (cont.)

ACCESS PARAMETERS -- how to read the file

RECORDS DELIMITED BY NEWLINE -- how to identify the start of a new row

FIELDS – the field name and datatype specifications

LOCATION – name of the actual file containing the data

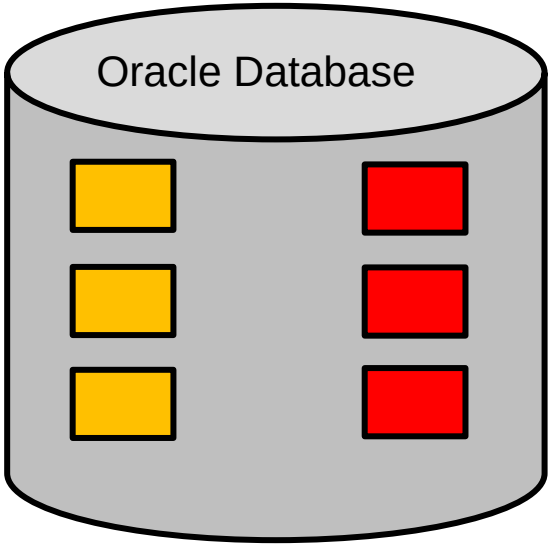
An example of the new syntax is found in red on the next slide.

External Tables Example

```
CREATE TABLE emp_load
  (employee_number CHAR(5),
   employee_dob CHAR(20),
   employee_last_name CHAR(20),
   employee_first_name CHAR(15),
   employee_middle_name CHAR(15),
   employee_hire_date DATE)
ORGANIZATION EXTERNAL
   (TYPE ORACLE_LOADER
    DEFAULT DIRECTORY def_dir1
    ACCESS PARAMETERS
   (RECORDS DELIMITED BY NEWLINE
    FIELDS (employee_number CHAR(2),
                    employee_dob CHAR(20),
                    employee_last_name CHAR(18),
                    employee_first_name CHAR(11),
                    employee_middle_name CHAR(11),
                    employee_hire_date CHAR(10) date_format DATE mask
"mm/dd/yyyy“))
   LOCATION ('info.dat'));
```

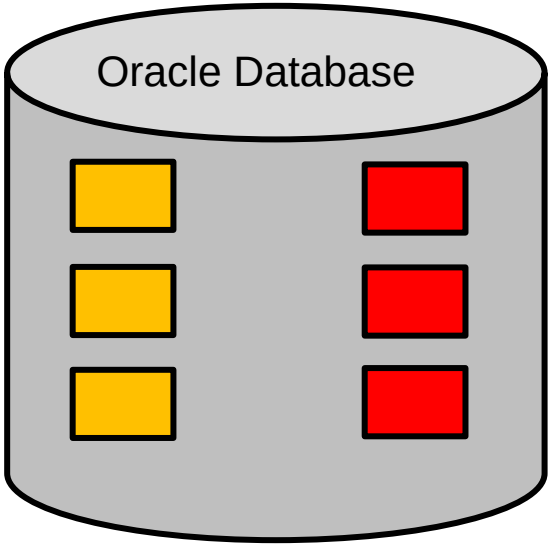
Data Dictionary

Two kinds of tables exists in an Oracle Database: User tables and Data Dictionary tables. You can issue SQL statements to access both kinds of tables—you can select, insert, update, and delete data in the user tables, and you can select data in the Data Dictionary tables.



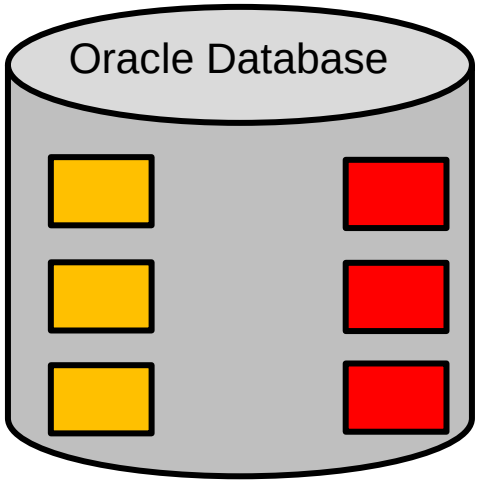
Data Dictionary (cont.)

1. User tables created by you containing your data:
   - D_CDS, D_SONGS, D_EVENTS, etc.
2. Data Dictionary tables:
   - DICTIONARY, USER_OBJECTS, USER_TABLES, USER_SEGMENTS, USER_INDEXES, etc.



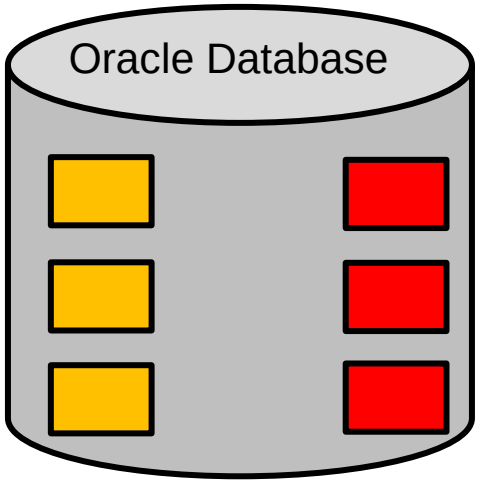
Data Dictionary (cont.)

The Data Dictionary tables are all owned by a special Oracle user called SYS and only SELECT statements should be used when working with any of these tables. To make these tables safe from accidental user access, they all have views created through which the Data Dictionary is accessed by database users.



Data Dictionary (cont.)

If any Oracle user attempts to do inserts, updates, or deletes against any of the Data Dictionary tables, the operation is disallowed as it might compromise the integrity of the entire database.



Data Dictionary (cont.)

When you are using the Data Dictionary views in the SQL Commands interface, you need to know the names of the Dictionary views you are working with. In Oracle, this is quite simple: prefix the object type you are looking for with a USER_xxx or an ALL_xxx, where xxx is the object type.

Data Dictionary (cont.)

So if you want to investigate indexes, then simply select from USER_INDEXES; if you want information about sequences, then the table is USER_SEQUENCES and so on.

Two very important views to remember are DICTIONARY (DICT) and DICT_COLUMNS.

Terminology

Key terms used in this lesson included:

- CREATE TABLE
- Data dictionary
- Table
- Schema
- DEFAULT

Summary

In this lesson, you should have learned how to:

- Categorize the main database objects
- Review a table structure
- Describe how schema objects are used
- List and provide an example of each of the number, date, and character data types

Summary (cont.)

In this lesson, you should have learned how to:

- Create a table using the appropriate data type for each column
- Explain the use of external tables
- Use the Data Dictionary to obtain the names and other attributes of database objects