Database Objects

An Oracle database can contain multiple data structures. Each structure should be outlined in the database design so that it can be created during the build stage of database development.

- Table: Stores data
- View: Subset of data from one or more tables
- Sequence: Numeric value generator
- Index: Improves the performance of some queries
- Synonym: Gives alternative names to objects

Oracle9*i* Table Structures

- Tables can be created at any time, even while users are using the database.
- You do not need to specify the size of any table. The size is ultimately defined by the amount of space allocated to the database as a whole. It is important, however, to estimate how much space a table will use over time.
- Table structure can be modified online.

Naming Rules

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, _, $, and #**
- **Must not duplicate the name of another object owned by the same user**
- **Must not be an Oracle Server reserved word**

Use descriptive names for tables and other database objects.

Note: Names are case insensitive. For example, `EMPLOYEES` is treated as the same name as `eMPloyees` or `eMpLOYEES`.

The `CREATE TABLE` Statement

- **You must have:**
  - **`CREATE TABLE` privilege**
  - **A storage area**

```
CREATE TABLE [schema.]table
             (column datatype [DEFAULT expr][, ...]);
```

- **You specify:**
  - **Table name**
  - **Column name, column data type, and column size**

Create tables to store data by executing the SQL CREATE TABLE statement. This statement is one of the data definition language (DDL) statements, which are covered in subsequent lessons. DDL statements are a subset of SQL statements used to create, modify, or remove Oracle9*i* database structures. These statements have an immediate effect on the database, and they also record information in the data dictionary.

To create a table, a user must have the CREATE TABLE privilege and a storage area in which to create objects. The database administrator uses data control language (DCL) statements, which are covered in a later lesson, to grant privileges to users.

In the syntax:

schema	is the same as the owner's name
table	is the name of the table
DEFAULT *expr*	specifies a default value if a value is omitted in the INSERT statement
column	is the name of the column
datatype	is the column's data type and length

Referencing Another User's Tables

- Tables belonging to other users are not in the user's schema.
- You should use the owner's name as a prefix to those tables.

A *schema* is a collection of objects. Schema objects are the logical structures that directly refer to the data in a database. Schema objects include tables, views, synonyms, sequences, stored procedures, indexes, clusters, and database links.

If a table does not belong to the user, the owner's name must be prefixed to the table. For example, if there is a schema named USER_B, and USER_B has an EMPLOYEES table, then specify the following to retrieve data from that table:

```
SELECT *
FROM user_b.employees;
```

The DEFAULT Option

- Specify a default value for a column during an INSERT operation.

```
... hire_date DATE DEFAULT SYSDATE, ...
```

- Literal values, expressions, or SQL functions are legal values.
- Another column's name or a pseudocolumn are illegal values.
- The default data type must match the column data type.

A column can be given a default value by using the DEFAULT option. This option prevents null values from entering the columns if a row is inserted without a value for the column. The default value can be a literal value, an expression, or a SQL function, such as SYSDATE and USER, but the value cannot be the name of another column or a pseudocolumn, such as NEXTVAL or CURRVAL. The default expression must match the data type of the column.

Note: CURRVAL and NEXTVAL are explained later.

Creating Tables

- **Create the table.**

```
CREATE TABLE dept
            (deptno  NUMBER(2),
             dname   VARCHAR2(14),
             loc     VARCHAR2(13));
Table created.
```

- **Confirm creation of the table.**

```
DESCRIBE dept
```

Name	Null?	Type
DEPTNO		NUMBER(2)
DNAME		VARCHAR2(14)
LOC		VARCHAR2(13)

The example on the slide creates the DEPT table, with three columns: namely, DEPTNO, DNAME, and LOC. It further confirms the creation of the table by issuing the DESCRIBE command.

Because creating a table is a DDL statement, an automatic commit takes place when this statement is executed.

Tables in the Oracle Database

- **User tables:**
  - **Are a collection of tables created and maintained by the user**
  - **Contain user information**
- **Data dictionary:**
  - **Is a collection of tables created and maintained by the Oracle Server**
  - **Contain database information**

User tables are tables created by the user, such as EMPLOYEES. There is another collection of tables and views in the Oracle Database known as the *data dictionary*. This collection is created and maintained by the Oracle Server and contains information about the database.

All data dictionary tables are owned by the SYS user. The base tables are rarely accessed by the user because the information in them is not easy to understand. Therefore, users typically access data dictionary views because the information is presented in a format that is easier to understand. Information stored in the data dictionary includes names of the Oracle Server users, privileges granted to users, database object names, table constraints, and auditing information.

There are four categories of data dictionary views; each category has a distinct prefix that reflects its intended use.

Prefix	Description
USER_	These views contain information about objects owned by the user.
ALL_	These views contain information about all of the tables (object tables and relational tables) accessible to the user.
DBA_	These views are restricted views, which can be accessed only by people who have been assigned the DBA role.
V$	These views are dynamic performance views, database server performance, memory, and locking.

Querying the Data Dictionary

- **See the names of tables owned by the user.**

```
SELECT   table_name
FROM user_tables;
```

- **View distinct object types owned by the user.**

```
SELECT DISTINCT object_type
FROM user_objects;
```

- **View tables, views, synonyms, and sequences owned by the user.**

```
SELECT *
FROM user_catalog;
```

You can query the data dictionary tables to view various database objects owned by you. The data dictionary tables frequently used are these:

- USER_TABLES
- USER_OBJECTS
- USER_CATALOG

Note: USER_CATALOG has a synonym called CAT. You can use this synonym instead of USER_CATALOG in SQL statements.

```
SELECT   *
FROM     CAT;
```

Data Types

Data type	Description
VARCHAR2(*size*)	Variable-length character data (a maximum *size* must be specified: Minimum *size* is 1; maximum *size* is 4000)
CHAR [(*size*)]	Fixed-length character data of length *size* bytes (default and minimum *size* is 1; maximum *size* is 2000)
NUMBER [(*p*,*s*)]	Number having precision *p* and scale *s* (The precision is the total number of decimal digits, and the scale is the number of digits to the right of the decimal point; the precision can range from 1 to 38 and the scale can range from -84 to 127)
DATE	Date and time values to the nearest second between January 1, 4712 B.C., and A.D. December 31, 9999
LONG	Variable-length character data up to 2 gigabytes
CLOB	Character data up to 4 gigabytes

Data type	Description
RAW(*size*)	Raw binary data of length *size* (a maximum *size* must be specified. maximum *size* is 2000)
LONG RAW	Raw binary data of variable length up to 2 gigabytes
BLOB	Binary data up to 4 gigabytes
BFILE	Binary data stored in an external file; up to 4 gigabytes
ROWID	Hexadecimal string representing the unique address of a row in its table. This datatype is primarily for values returned by the ROWID pseudocolumn.

- A LONG column is not copied when a table is created using a subquery.
- A LONG column cannot be included in a GROUP BY or an ORDER BY clause.
- Only one LONG column can be used per table.
- No constraints can be defined on a LONG column.
- You may want to use a CLOB column rather than a LONG column.

Datetime Data Types

Datetime enhancements with Oracle9*i*:

- **New datetime data types have been introduced.**
- **New data type storage is available.**
- **Enhancements have been made to time zones and local time zone.**

Data Type	Description
TIMESTAMP	Allows the time to be stored as a date with fractional seconds. There are several variations of the data type.
INTERVAL YEAR TO MONTH	Allows time to stored as an interval of years and months.
INTERVAL DAY TO SECOND	Allows time to be stored as an interval of days to hours minutes and seconds.

- **The `TIMESTAMP` data type is an extension of the `DATE` data type.**
- **It stores the year, month, and day of the `DATE` data type, plus hour, minute, and second values as well as the fractional second value.**
- **The `TIMESTAMP` data type is specified as follows:**

```
TIMESTAMP[(fractional_seconds_precision)]
```

The `fractional_seconds_precision` optionally specifies the number of digits in the fractional part of the `SECOND` datetime field and can be a number in the range 0 to 9. The default is 6.

Example

```
CREATE TABLE new_employees
(employee_id NUMBER,
 first_name VARCHAR2(15),
 last_name VARCHAR2(15),
 ...
 start_date TIMESTAMP(7),
 ...);
```

In the preceding example, we created a table `NEW_EMPLOYEES` with a column `start_date` with a data type of `TIMESTAMP`. The precision of `'7'` indicates the fractional seconds precision which if not specified defaults to `'6'`.

Assume that two rows are inserted into the `NEW_EMPLOYEES` table. The output shows the differences in the display. (A `DATE` data type defaults to display the format of `DD-MON-RR`):

```
SELECT start_date FROM new_employees;

17-JUN-87 12.00.00.000000 AM
21-SEP-89 12.00.00.000000 AM
```

TIMESTAMP WITH TIME ZONE Data Type

- **TIMESTAMP WITH TIME ZONE is a variant of TIMESTAMP that includes a time zone displacement in its value.**
- **The time zone displacement is the difference, in hours and minutes, between local time and UTC.**

```
TIMESTAMP[(fractional_seconds_precision)]
WITH TIME ZONE
```

UTC stand for Coordinated Universal Time: formerly Greenwich Mean Time. Two `TIMESTAMP WITH TIME ZONE` values are considered identical if they represent the same instant in UTC, regardless of the `TIME ZONE` offsets stored in the data.

For example,

```
TIMESTAMP '1999-04-15 8:00:00 -8:00'
```

is the same as

```
TIMESTAMP '1999-04-15 11:00:00 -5:00'
```

That is, 8:00 a.m. Pacific Standard Time is the same as 11:00 a.m. Eastern Standard Time.

This can also be specified as

```
TIMESTAMP '1999-04-15 8:00:00 US/Pacific'
```

Note: `fractional_seconds_precision` optionally specifies the number of digits in the fractional part of the `SECOND` datetime field and can be a number in the range 0 to 9. The default is 6.

TIMESTAMP WITH LOCAL TIME Data Type

- **TIMESTAMP WITH LOCAL TIME ZONE is another variant of TIMESTAMP that includes a time zone displacement in its value.**
- **Data stored in the database is normalized to the database time zone**
- **The time zone displacement is not stored as part of the column data; the server returns the data in the users' local session time zone.**
- **TIMESTAMP WITH LOCAL TIME ZONE data type is specified as follows:**

```
TIMESTAMP[(fractional_seconds_precision)]
WITH LOCAL TIME ZONE
```

Example

```
CREATE TABLE time_example as (order_date TIMESTAMP
WITH LOCAL TIME ZONE);

INSERT INTO time_example VALUES('15-NOV-00 09:34:28 AM');

SELECT * FROM time_example;

order_date
---------------------
15-NOV-00 09.34.28 AM
```

INTERVAL YEAR TO MONTH Data Type

- **INTERVAL YEAR TO MONTH stores a period of time using the YEAR and MONTH datetime fields.**

```
INTERVAL YEAR [(year_precision)] TO MONTH
```

- **Example:**

```
INTERVAL '312-2' YEAR(3) TO MONTH
Indicates an interval of 312 years and 2 months

INTERVAL '312' YEAR(3)
Indicates 312 years and 0 months

INTERVAL '300' MONTH(3)
Indicates an interval of 300 months
```

Restriction

The leading field must be more significant than the trailing field. For example, `INTERVAL '0-1 MONTH TO YEAR` is not valid.

Creating a Table by Using a Subquery Syntax

- Create a table and insert rows by combining the `CREATE TABLE` statement and the `AS` *`subquery`* option.

```
CREATE TABLE table
       [(column, column...)]
AS subquery;
```

- Match the number of specified columns to the number of subquery columns.
- Define columns with column names and default values.

Guidelines

- The table is created with the specified column names, and the rows retrieved by the SELECT statement are inserted into the table.
- The column definition can contain only the column name and default value.
- If column specifications are given, the number of columns must equal the number of columns in the subquery SELECT list.
- If no column specifications are given, the column names of the table are the same as the column names in the subquery.
- The integrity rules are not passed onto the new table, only the column data type definitions.

Creating a Table by Using a Subquery

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

```
DESCRIBE dept80
```

Name	Null?	Type
EMPLOYEE_ID		NUMBER(6)
LAST_NAME	NOT NULL	VARCHAR2(25)
ANNSAL		NUMBER
HIRE_DATE	NOT NULL	DATE

The ALTER TABLE Statement

Use the ALTER TABLE statement to:

- Add a new column
- Modify an existing column
- Define a default value for the new column
- Drop a column

Use the ALTER TABLE statement to add, modify or drop columns.

```
ALTER TABLE table
ADD         (column datatype [DEFAULT expr]
            [, column datatype]...);
```

```
ALTER TABLE table
MODIFY      (column datatype [DEFAULT expr]
            [, column datatype]...);
```

```
ALTER TABLE table
DROP        (column);
```

Adding a Column

New column

DEPT80

EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE
149	Zlotkey	126000	29-JAN-00
174	Abel	132000	11-MAY-96
176	Taylor	103200	24-MAR-98

JOB_ID

Add a new column to the DEPT80 table.

DEPT80

EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE	JOB_ID
149	Zlotkey	126000	29-JAN-00	
174	Abel	132000	11-MAY-96	
176	Taylor	103200	24-MAR-98	

The graphic in the slide adds the JOB_ID column to the DEPT80 table. Notice that the new column becomes the last column in the table.

- **Use the ADD clause to add columns.**

```
ALTER TABLE dept80
ADD         (job_id VARCHAR2(9));
Table altered.
```

- **The new column becomes the last column.**

EMPLOYEE_ID	LAST_NAME	ANNSAL	HIRE_DATE	JOB_ID
149	Zlotkey	126000	29-JAN-00	
174	Abel	132000	11-MAY-96	
176	Taylor	103200	24-MAR-98	

Guidelines for Adding a Column

- You can add or modify columns.
- You cannot specify where the column is to appear. The new column becomes the last column.

The example in the slide adds a column named JOB_ID to the DEPT80 table. The JOB_ID column becomes the last column in the table.

Note: If a table already contains rows when a column is added, then the new column is initially null for all the rows.

Modifying a Column

- **You can change a column's data type, size, and default value.**

```
ALTER TABLE   dept80
MODIFY        (last_name VARCHAR2(30));
Table altered.
```

- **A change to the default value affects only subsequent insertions to the table.**

Guidelines

- You can increase the width or precision of a numeric column.
- You can increase the width of numeric or character columns.
- You can decrease the width of a column only if the column contains only null values or if the table has no rows.
- You can change the data type only if the column contains null values.
- You can convert a CHAR column to the VARCHAR2 data type or convert a VARCHAR2 column to the CHAR data type only if the column contains null values or if you do not change the size.
- A change to the default value of a column affects only subsequent insertions to the table.

Dropping a Column

Use the DROP COLUMN clause to drop columns you no longer need from the table.

```
ALTER TABLE   dept80
DROP  COLUMN   job_id;
Table altered.
```

Guidelines

- The column may or may not contain data.
- Using the ALTER TABLE statement, only one column can be dropped at a time.
- The table must have at least one column remaining in it after it is altered.
- Once a column is dropped, it cannot be recovered.

The SET UNUSED Option

- You use the SET UNUSED option to mark one or more columns as unused.
- You use the DROP UNUSED COLUMNS option to remove the columns that are marked as unused.

```
ALTER TABLE    table
SET   UNUSED  (column);
           OR
ALTER       TABLE  table
SET UNUSED COLUMN column;
```

```
ALTER TABLE table
DROP  UNUSED COLUMNS;
```

The SET UNUSED Option

The SET UNUSED option marks one or more columns as unused so that they can be dropped when the demand on system resources is lower. This is a feature available in Oracle8*i* and later release. Specifying this clause does not actually remove the target columns from each row in the table (that is, it does not restore the disk space used by these columns). Therefore, the response time is faster than if you executed the DROP clause. Unused columns are treated as if they were dropped, even though their column data remains in the table's rows. After a column has been marked as unused, you have no access to that column. A SELECT * query will not retrieve data from unused columns. In addition, the names and types of columns marked unused will not be displayed during a DESCRIBE, and you can add to the table a new column with the same name as an unused column. SET UNUSED information is stored in the USER_UNUSED_COL_TABS dictionary view.

The DROP UNUSED COLUMNS Option

DROP UNUSED COLUMNS removes from the table all columns currently marked as unused. You can use this statement when you want to reclaim the extra disk space from unused columns in the table. If the table contains no unused columns, the statement returns with no errors.

```
ALTER TABLE  dept80
SET UNUSED (last_name);
Table altered.

ALTER TABLE  dept80
DROP  UNUSED COLUMNS;
Table altered.
```

Dropping a Table

- All data and structure in the table is deleted.
- Any pending transactions are committed.
- All indexes are dropped.
- You *cannot* roll back the DROP TABLE statement.

```
DROP TABLE dept80;
Table dropped.
```

Guidelines

- All data is deleted from the table.
- Any views and synonyms remain but are invalid.
- Any pending transactions are committed.
- Only the creator of the table or a user with the DROP ANY TABLE privilege can remove a table.

Note: The DROP TABLE statement, once executed, is irreversible. The Oracle Server does not question the action when you issue the DROP TABLE statement. If you own that table or have a high-level privilege, then the table is immediately removed. As with all DDL statements, DROP TABLE is committed automatically.

Changing the Name of an Object

- To change the name of a table, view, sequence, or synonym, execute the RENAME statement.

```
RENAME dept TO detail_dept;
Table renamed.
```

- You must be the owner of the object.

Truncating a Table

- The TRUNCATE TABLE statement:
  - Removes all rows from a table
  - Releases the storage space used by that table

```
TRUNCATE TABLE detail_dept;
Table truncated.
```

- You cannot roll back row removal when using TRUNCATE.
- Alternatively, you can remove rows by using the DELETE statement.

The DELETE statement can also remove all rows from a table, but it does not release storage space. The TRUNCATE command is faster. Removing rows with the TRUNCATE statement is faster than removing them with the DELETE statement for the following reasons:

- The TRUNCATE statement is a data definition language (DDL) statement and generates no rollback information.
- Truncating a table does not fire the delete triggers of the table.
- If the table is the parent of a referential integrity constraint, you cannot truncate the table. Disable the constraint before issuing the TRUNCATE statement.

Adding Comments to a Table

- You can add comments to a table or column by using the COMMENT statement.

```
COMMENT ON TABLE employees
IS 'Employee Information';
Comment created.
```

- Comments can be viewed through the data dictionary views:
  - ALL_COL_COMMENTS
  - USER_COL_COMMENTS
  - ALL_TAB_COMMENTS
  - USER_TAB_COMMENTS

You can add a comment of up to 2,000 bytes about a column, table, view, or snapshot by using the COMMENT statement. The comment is stored in the data dictionary and can be viewed in one of the following data dictionary views in the COMMENTS column:

- ALL_COL_COMMENTS
- USER_COL_COMMENTS
- ALL_TAB_COMMENTS
- USER_TAB_COMMENTS

Syntax

```
COMMENT ON TABLE table | COLUMN table.column
  IS 'text';
```

In the syntax:

table	is the name of the table
column	is the name of the column in a table
text	is the text of the comment

You can drop a comment from the database by setting it to empty string (''):

```
COMMENT ON TABLE  employees IS ' ';
```

ASSIGNMENTS

1. Create the DEPT table based on the following table instance chart. Place the syntax in a script called lab9_1.sql, then execute the statement in the script to create the table. Confirm that the table is created.

Column Name	ID	NAME
Key Type		
Nulls/Unique		
FK Table		
FK Column		
Data type	NUMBER	VARCHAR2
Length	7	25

Name	Null?	Type
ID		NUMBER(7)
NAME		VARCHAR2(25)

2. Populate the DEPT table with data from the DEPARTMENTS table. Include only columns that you need.
3. Create the EMP table based on the following table instance chart. Place the syntax in a script called lab9_3.sql, and then execute the statement in the script to create the table. Confirm that the table is created.

Column Name	ID	LAST_NAME	FIRST_NAME	DEPT_ID
Key Type				
Nulls/Unique				
FK Table				
FK Column				
Data type	NUMBER	VARCHAR2	VARCHAR2	NUMBER
Length	7	25	25	7

Name	Null?	Type
ID		NUMBER(7)
LAST_NAME		VARCHAR2(25)
FIRST_NAME		VARCHAR2(25)
DEPT_ID		NUMBER(7)

4. Modify the EMP table to allow for longer employee last names. Confirm your modification.

Name	Null?	Type
ID		NUMBER(7)
LAST_NAME		VARCHAR2(50)
FIRST_NAME		VARCHAR2(25)
DEPT_ID		NUMBER(7)

5. Confirm that both the DEPT and EMP tables are stored in the data dictionary. (**Hint:** USER_TABLES)

TABLE_NAME
DEPT
EMP

6. Create the EMPLOYEES2 table based on the structure of the EMPLOYEES table. Include only the EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY, and DEPARTMENT_ID columns. Name the columns in your new table ID, FIRST_NAME, LAST_NAME, SALARY , and DEPT_ID, respectively.
7. Drop the EMP table.
8. Rename the EMPLOYEES2 table as EMP.
9. Add a comment to the DEPT and EMP table definitions describing the tables. Confirm your additions in the data dictionary.
10. Drop the FIRST_NAME column from the EMP table. Confirm your modification by checking the description of the table.
11. In the EMP table, mark the DEPT_ID column in the EMP table as UNUSED. Confirm your modification by checking the description of the table.
12. Drop all the UNUSED columns from the EMP table. Confirm your modification by checking the description of the table.