Creating and Managing Tables
Objectives

After completing this lesson, you should be able to do the following:

• Describe the main database objects
• Create tables
• Describe the data types that can be used when specifying column definition
• Alter table definitions
• Drop, rename, and truncate tables
## Database Objects

<table>
<thead>
<tr>
<th>Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>Basic unit of storage; composed of rows and columns</td>
</tr>
<tr>
<td>View</td>
<td>Logically represents subsets of data from one or more tables</td>
</tr>
<tr>
<td>Sequence</td>
<td>Numeric value generator</td>
</tr>
<tr>
<td>Index</td>
<td>Improves the performance of some queries</td>
</tr>
<tr>
<td>Synonym</td>
<td>Gives alternative names to objects</td>
</tr>
</tbody>
</table>
Naming Rules

Table names and column names:

• Must begin with a letter
• Must be 1–30 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
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
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.
The `DEFAULT` Option

• Specify a default value for a column during an insert.

```sql
... 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.
Creating Tables

• Create the table.
CREATE TABLE dept
(deptno NUMBER(2),
dname VARCHAR2(14),
loc VARCHAR2(13));

Table created.

• Confirm table creation.

DESCRIBE dept

<table>
<thead>
<tr>
<th>Name</th>
<th>Null?</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEPTNO</td>
<td></td>
<td>NUMBER(2)</td>
</tr>
<tr>
<td>DNAME</td>
<td></td>
<td>VARCHAR2(14)</td>
</tr>
<tr>
<td>LOC</td>
<td></td>
<td>VARCHAR2(13)</td>
</tr>
</tbody>
</table>
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
Querying the Data Dictionary

• See the names of tables owned by the user.

```sql
SELECT table_name
FROM user_tables;
```

• View distinct object types owned by the user.

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

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

```sql
SELECT *
FROM user_catalog;
```
## Data Types

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARCHAR2(size)</td>
<td>Variable-length character data</td>
</tr>
<tr>
<td>CHAR(size)</td>
<td>Fixed-length character data</td>
</tr>
<tr>
<td>NUMBER(p,s)</td>
<td>Variable-length numeric data</td>
</tr>
<tr>
<td>DATE</td>
<td>Date and time values</td>
</tr>
<tr>
<td>LONG</td>
<td>Variable-length character data up to 2 gigabytes</td>
</tr>
<tr>
<td>CLOB</td>
<td>Character data up to 4 gigabytes</td>
</tr>
<tr>
<td>RAW and LONG RAW</td>
<td>Raw binary data</td>
</tr>
<tr>
<td>BLOB</td>
<td>Binary data up to 4 gigabytes</td>
</tr>
<tr>
<td>BFILE</td>
<td>Binary data stored in an external file; up to 4 gigabytes</td>
</tr>
<tr>
<td>ROWID</td>
<td>A 64 base number system representing the unique address of a row in its table.</td>
</tr>
</tbody>
</table>
DateTime Data Types

Datetime enhancements with Oracle9i:

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

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIMESTAMP</td>
<td>Date with fractional seconds</td>
</tr>
<tr>
<td>INTERVAL YEAR TO MONTH</td>
<td>Stored as an interval of years and months</td>
</tr>
<tr>
<td>INTERVAL DAY TO SECOND</td>
<td>Stored as an interval of days to hours minutes and seconds</td>
</tr>
</tbody>
</table>
DateTime Data Types

- 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)]`
**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
```
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; Oracle returns the data in the users' local session time zone.

- **TIMESTAMP WITH LOCAL TIME ZONE** data type is specified as follows:

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

INTERVAL '123-2' YEAR(3) TO MONTH
Indicates an interval of 123 years, 2 months.

INTERVAL '123' YEAR(3)
Indicates an interval of 123 years 0 months.

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

INTERVAL '123' YEAR
Returns an error, because the default precision is 2, and '123' has 3 digits.
INTERVAL DAY TO SECOND Data Type

- INTERVAL DAY TO SECOND stores a period of time in terms of days, hours, minutes, and seconds.

```
INTERVAL DAY [(day_precision)]
  TO SECOND [(fractional_seconds_precision)]
```

```
INTERVAL '4 5:12:10.222' DAY TO SECOND(3)
Indicates 4 days, 5 hours, 12 minutes, 10 seconds, and 222 thousandths of a second.
INTERVAL '123' YEAR(3).

INTERVAL '7' DAY
Indicates 7 days.

INTERVAL '180' DAY(3)
Indicates 180 days.
```
INTERVAL DAY TO SECOND Data Type

- INTERVAL DAY TO SECOND stores a period of time in terms of days, hours, minutes, and seconds.

```
INTERVAL '4 5:12:10.222' DAY TO SECOND(3)
Indicates 4 days, 5 hours, 12 minutes, 10 seconds, and 222 thousandths of a second.

INTERVAL '4 5:12' DAY TO MINUTE
Indicates 4 days, 5 hours and 12 minutes.

INTERVAL '400 5' DAY(3) TO HOUR
Indicates 400 days 5 hours.

INTERVAL '11:12:10.2222222' HOUR TO SECOND(7)
indicates 11 hours, 12 minutes, and 10.2222222 seconds.
```
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.

```sql
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.
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

<table>
<thead>
<tr>
<th>Name</th>
<th>Null?</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>EMPLOYEE_ID</td>
<td>NOT NULL</td>
<td>NUMBER(6)</td>
</tr>
<tr>
<td>LAST_NAME</td>
<td>NOT NULL</td>
<td>VARCHAR2(25)</td>
</tr>
<tr>
<td>ANNSAL</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>HIRE_DATE</td>
<td>NOT NULL</td>
<td>DATE</td>
</tr>
</tbody>
</table>
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
The **ALTER TABLE** Statement

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

### DEPT80

<table>
<thead>
<tr>
<th>EMPLOYEE_ID</th>
<th>LAST_NAME</th>
<th>ANNSAL</th>
<th>HIRE_DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>149</td>
<td>Zlotkey</td>
<td>126000</td>
<td>29-JAN-00</td>
</tr>
<tr>
<td>174</td>
<td>Abel</td>
<td>132000</td>
<td>11-MAY-96</td>
</tr>
<tr>
<td>176</td>
<td>Taylor</td>
<td>103200</td>
<td>24-MAR-98</td>
</tr>
</tbody>
</table>

### DEPT80

<table>
<thead>
<tr>
<th>EMPLOYEE_ID</th>
<th>LAST_NAME</th>
<th>ANNSAL</th>
<th>HIRE_DATE</th>
<th>JOB_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>149</td>
<td>Zlotkey</td>
<td>126000</td>
<td>29-JAN-00</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td>Abel</td>
<td>132000</td>
<td>11-MAY-96</td>
<td></td>
</tr>
<tr>
<td>176</td>
<td>Taylor</td>
<td>103200</td>
<td>24-MAR-98</td>
<td></td>
</tr>
</tbody>
</table>

“Add a new column to the DEPT80 table.”
Adding a Column

• You use the ADD clause to add columns.

```
ALTER TABLE dept80
    ADD (job_id VARCHAR2(9));
```

Table altered.

• The new column becomes the last column.

<table>
<thead>
<tr>
<th>EMPLOYEE_ID</th>
<th>LAST_NAME</th>
<th>SALARY</th>
<th>HIRE_DATE</th>
<th>JOB_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>149</td>
<td>Zlotkey</td>
<td>126000</td>
<td>29-JAN-00</td>
<td></td>
</tr>
<tr>
<td>174</td>
<td>Abel</td>
<td>132000</td>
<td>11-MAY-96</td>
<td></td>
</tr>
<tr>
<td>176</td>
<td>Taylor</td>
<td>103200</td>
<td>24-MAR-98</td>
<td></td>
</tr>
</tbody>
</table>
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.
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.
```
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.

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

ALTER TABLE table
DROP UNUSED COLUMNS;
```
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.
```
Changing the Name of an Object

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

  ```sql
  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

```sql
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.
Adding Comments to a Table

- You can add comments to a table or column by using the `COMMENT` statement.
  ```sql
  COMMENT ON TABLE employees IS 'Employee Information';
  ```

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

In this lesson, you should have learned how to use DDL statements to create, alter, drop, and rename tables.

<table>
<thead>
<tr>
<th>Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CREATE TABLE</td>
<td>Creates a table</td>
</tr>
<tr>
<td>ALTER TABLE</td>
<td>Modifies table structures</td>
</tr>
<tr>
<td>DROP TABLE</td>
<td>Removes the rows and table structure</td>
</tr>
<tr>
<td>RENAME</td>
<td>Changes the name of a table, view, sequence, or synonym</td>
</tr>
<tr>
<td>TRUNCATE</td>
<td>Removes all rows from a table and releases the storage space</td>
</tr>
<tr>
<td>COMMENT</td>
<td>Adds comments to a table or view</td>
</tr>
</tbody>
</table>
Practice 9 Overview

This practice covers the following topics:

• Creating new tables
• Creating a new table by using the `CREATE TABLE AS` syntax
• Modifying column definitions
• Verifying that the tables exist
• Adding comments to tables
• Dropping tables
• Altering tables