



# Using DDL Statements to Create and Manage Tables

# Objectives

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

- **Categorize the main database objects**
- **Review the table structure**
- **List the data types that are available for columns**
- **Create a simple table**
- **Understand how constraints are created at the time of table creation**
- **Describe how schema objects work**

# Database Objects

<b>Object</b>	<b>Description</b>
<b>Table</b>	<b>Basic unit of storage; composed of rows</b>
<b>View</b>	<b>Logically represents subsets of data from one or more tables</b>
<b>Sequence</b>	<b>Generates numeric values</b>
<b>Index</b>	<b>Improves the performance of some queries</b>
<b>Synonym</b>	<b>Gives alternative names to objects</b>

# 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**

# CREATE TABLE Statement

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

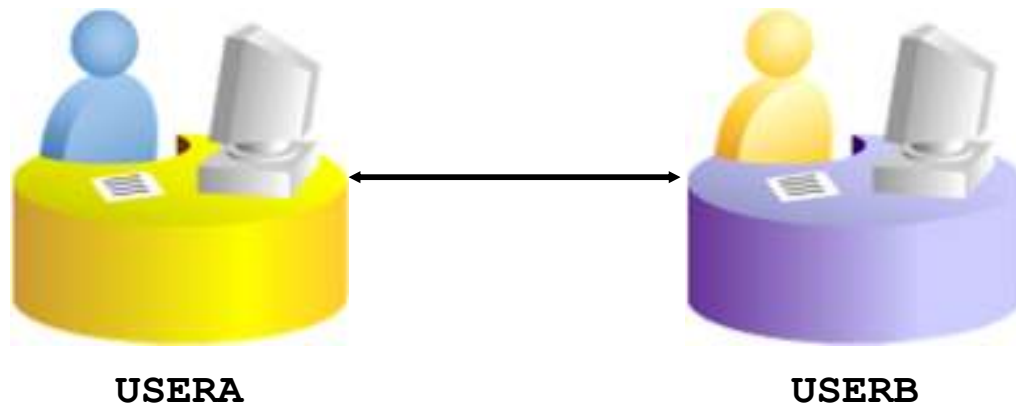
```
CREATE TABLE [schema.]table  
            (column datatype [DEFAULT expr] [, ...]);
```

- 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.



```
SELECT *  
FROM userB.employees;
```

```
SELECT *  
FROM userA.employees;
```

# DEFAULT Option

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

```
... 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.

```
CREATE TABLE hire_dates  
  (id          NUMBER(8),  
   hire_date DATE DEFAULT SYSDATE);
```

Table created.

# Creating Tables

- **Create the table.**

```
CREATE TABLE dept
  (deptno      NUMBER(2) ,
   dname       VARCHAR2(14) ,
   loc         VARCHAR2(13) ,
   create_date DATE DEFAULT SYSDATE) ;
```

Table created.

- **Confirm table creation.**

```
DESCRIBE dept
```

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



# Data Types

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

# Datetime Data Types

You can use several datetime data types:

Data Type	Description
<b>TIMESTAMP</b>	<b>Date with fractional seconds</b>
<b>INTERVAL YEAR TO MONTH</b>	<b>Stored as an interval of years and months</b>
<b>INTERVAL DAY TO SECOND</b>	<b>Stored as an interval of days, hours, minutes, and seconds</b>



# 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.
- You can optionally specify the time zone.

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

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

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

# Datetime Data Types

- The **INTERVAL YEAR TO MONTH** data type stores a period of time using the **YEAR** and **MONTH** datetime fields:

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

- The **INTERVAL DAY TO SECOND** data type stores a period of time in terms of days, hours, minutes, and seconds:

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

# 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.

# Including Constraints

- **Constraints enforce rules at the table level.**
- **Constraints prevent the deletion of a table if there are dependencies.**
- **The following constraint types are valid:**
  - NOT NULL
  - UNIQUE
  - PRIMARY KEY
  - FOREIGN KEY
  - CHECK



# Constraint Guidelines

- You can name a constraint, or the Oracle server generates a name by using the `SYS_Cn` format.
- Create a constraint at either of the following times:
  - At the same time as the table is created
  - After the table has been created
- Define a constraint at the column or table level.
- View a constraint in the data dictionary.

# Defining Constraints

- **Syntax:**

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

- **Column-level constraint:**

```
column [CONSTRAINT constraint_name] constraint_type,
```

- **Table-level constraint:**

```
column, ...  
    [CONSTRAINT constraint_name] constraint_type  
    (column, ...),
```



# Defining Constraints

- **Column-level constraint:**

```
CREATE TABLE employees(  
  employee_id NUMBER(6)  
    CONSTRAINT emp_emp_id_pk PRIMARY KEY,  
  first_name VARCHAR2(20),  
  ...);
```

1

- **Table-level constraint:**

```
CREATE TABLE employees(  
  employee_id NUMBER(6),  
  first_name VARCHAR2(20),  
  ...  
  job_id VARCHAR2(10) NOT NULL,  
  CONSTRAINT emp_emp_id_pk  
    PRIMARY KEY (EMPLOYEE_ID));
```

2

# NOT NULL Constraint

Ensures that null values are not permitted for the column:

EMPLOYEE_ID	LAST_NAME	EMAIL	PHONE_NUMBER	HIRE_DATE	JOB_ID	SALARY	DEPARTMENT_ID
100	King	SKING	515.123.4567	17-JUN-87	AD_PRES	24000	90
101	Kochhar	NKOCHHAR	515.123.4568	21-SEP-89	AD_VP	17000	90
102	De Haan	LDEHAAN	515.123.4569	13-JAN-93	AD_VP	17000	90
103	Hunold	AHUNOLD	590.423.4567	03-JAN-90	IT_PROG	9000	60
104	Ernst	BERNST	590.423.4568	21-MAY-91	IT_PROG	6000	60
178	Grant	KGRANT	011.44.1644.429263	24-MAY-99	SA_REP	7000	
200	Whalen	JWHALEN	515.123.4444	17-SEP-87	AD_ASST	4400	10

...  
20 rows selected.

↑  
**NOT NULL constraint**  
(No row can contain a null value for this column.)

↑  
**NOT NULL constraint**

↑  
**Absence of NOT NULL constraint**  
(Any row can contain a null value for this column.)

# UNIQUE Constraint

**EMPLOYEES**

EMPLOYEE_ID	LAST_NAME	EMAIL
100	King	SKING
101	Kochhar	NKOCHHAR
102	De Haan	LDEHAAN
103	Hunold	AHUNOLD
104	Ernst	BERNST

UNIQUE constraint

...

↑ INSERT INTO

208	Smith	JSMITH
209	Smith	JSMITH

← Allowed  
← Not allowed:  
already exists

# UNIQUE Constraint

Defined at either the table level or the column level:

```
CREATE TABLE employees (  
    employee_id      NUMBER(6),  
    last_name        VARCHAR2(25) NOT NULL,  
    email            VARCHAR2(25),  
    salary           NUMBER(8,2),  
    commission_pct   NUMBER(2,2),  
    hire_date        DATE NOT NULL,  
    ...  
    CONSTRAINT emp_email_uk UNIQUE(email));
```

# PRIMARY KEY Constraint

DEPARTMENTS

PRIMARY KEY

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500

...

Not allowed  
(null value)

↑ INSERT INTO

	Public Accounting		1400
50	Finance	124	1500

Not allowed  
(50 already exists)

# FOREIGN KEY Constraint

**DEPARTMENTS**

DEPARTMENT_ID	DEPARTMENT_NAME	MANAGER_ID	LOCATION_ID
10	Administration	200	1700
20	Marketing	201	1800
50	Shipping	124	1500
60	IT	103	1400
80	Sales	149	2500

**PRIMARY KEY** →

...

**EMPLOYEES**

EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
100	King	90
101	Kochhar	90
102	De Haan	90
103	Hunold	60
104	Ernst	60
107	Lorentz	60

← **FOREIGN KEY**

...

↑ **INSERT INTO**

200	Ford	9
201	Ford	60

← **Not allowed (9 does not exist)**

← **Allowed**

# FOREIGN KEY Constraint

Defined at either the table level or the column level:

```
CREATE TABLE employees(  
    employee_id      NUMBER(6),  
    last_name        VARCHAR2(25) NOT NULL,  
    email            VARCHAR2(25),  
    salary           NUMBER(8,2),  
    commission_pct   NUMBER(2,2),  
    hire_date        DATE NOT NULL,  
    ...  
    department_id    NUMBER(4),  
    CONSTRAINT emp_dept_fk FOREIGN KEY (department_id)  
        REFERENCES departments(department_id),  
    CONSTRAINT emp_email_uk UNIQUE(email));
```

# FOREIGN KEY Constraint: Keywords

- **FOREIGN KEY:** Defines the column in the child table at the table-constraint level
- **REFERENCES:** Identifies the table and column in the parent table
- **ON DELETE CASCADE:** Deletes the dependent rows in the child table when a row in the parent table is deleted
- **ON DELETE SET NULL:** Converts dependent foreign key values to null



# CHECK Constraint

- **Defines a condition that each row must satisfy**
- **The following expressions are not allowed:**
  - **References to CURRVAL, NEXTVAL, LEVEL, and ROWNUM pseudocolumns**
  - **Calls to SYSDATE, UID, USER, and USERENV functions**
  - **Queries that refer to other values in other rows**

```
..., salary NUMBER(2)  
      CONSTRAINT emp_salary_min  
      CHECK (salary > 0), ...
```

# CREATE TABLE: Example

```
CREATE TABLE employees
  ( employee_id      NUMBER(6)
    CONSTRAINT      emp_employee_id  PRIMARY KEY
  , first_name      VARCHAR2(20)
  , last_name       VARCHAR2(25)
    CONSTRAINT      emp_last_name_nn NOT NULL
  , email           VARCHAR2(25)
    CONSTRAINT      emp_email_nn     NOT NULL
    CONSTRAINT      emp_email_uk     UNIQUE
  , phone_number    VARCHAR2(20)
  , hire_date       DATE
    CONSTRAINT      emp_hire_date_nn NOT NULL
  , job_id          VARCHAR2(10)
    CONSTRAINT      emp_job_nn       NOT NULL
  , salary          NUMBER(8,2)
    CONSTRAINT      emp_salary_ck    CHECK (salary>0)
  , commission_pct  NUMBER(2,2)
  , manager_id      NUMBER(6)
  , department_id   NUMBER(4)
    CONSTRAINT      emp_dept_fk      REFERENCES
    departments (department_id));
```

# Violating Constraints

```
UPDATE employees
SET    department_id = 55
WHERE  department_id = 110;
```

```
UPDATE employees
      *
ERROR at line 1:
ORA-02291: integrity constraint (HR.EMP_DEPT_FK)
violated - parent key not found
```

**Department 55 does not exist.**

# Violating Constraints

**You cannot delete a row that contains a primary key that is used as a foreign key in another table.**

```
DELETE FROM departments
WHERE      department_id = 60;
```

```
DELETE FROM departments
          *
ERROR at line 1:
ORA-02292: integrity constraint (HR.EMP_DEPT_FK)
violated - child record found
```

# Creating a Table by Using a Subquery

- 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.

# 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

# 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

# Dropping a Table

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

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



# Summary

**In this lesson, you should have learned how to use the `CREATE TABLE` statement to create a table and include constraints.**

- **Categorize the main database objects**
- **Review the table structure**
- **List the data types that are available for columns**
- **Create a simple table**
- **Understand how constraints are created at the time of table creation**
- **Describe how schema objects work**