# 9

# Introduction to Data Definition Language

# 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
- Explain how constraints are created at the time of table creation

# Course Roadmap

Lesson 1: Introduction

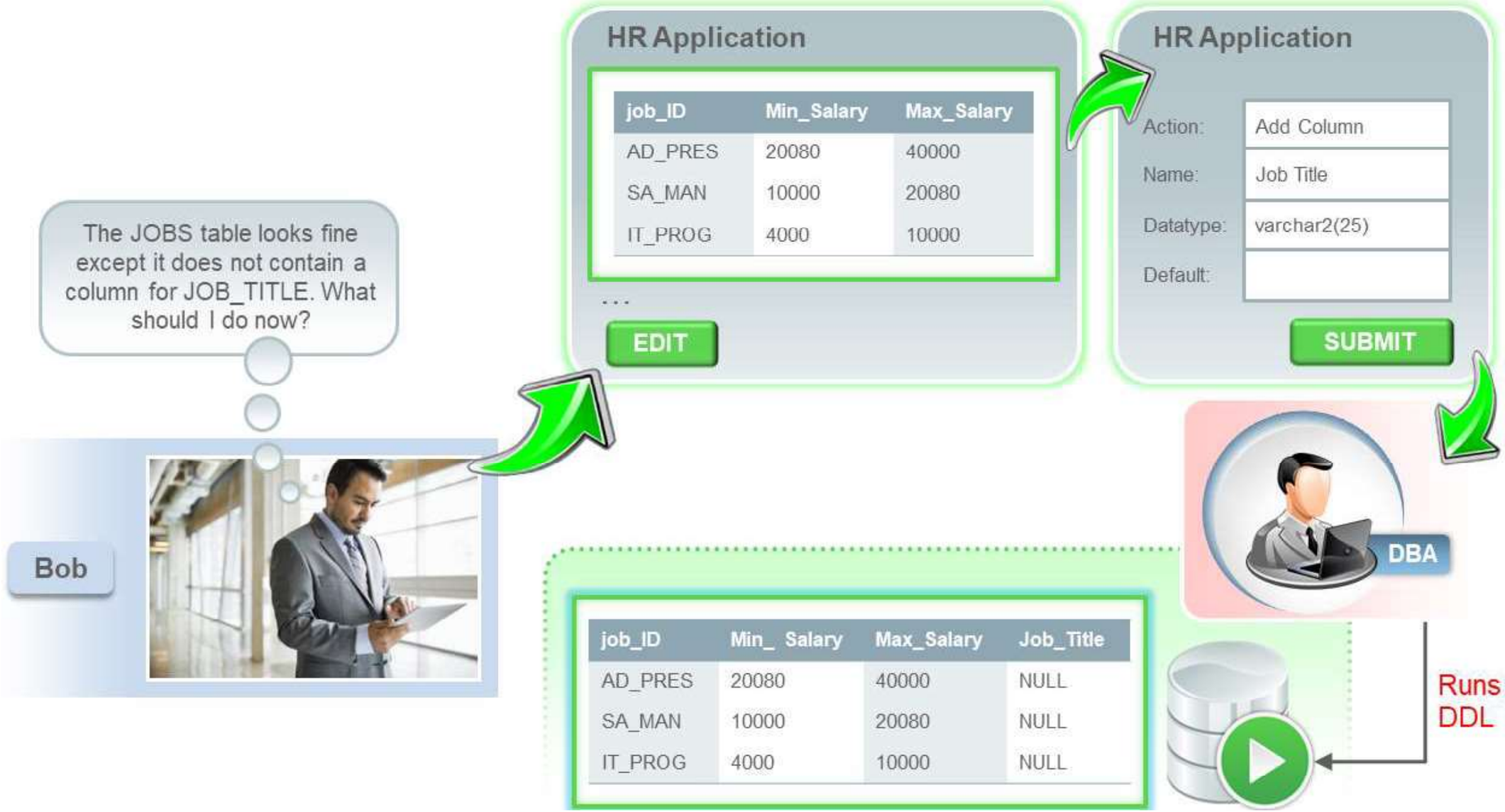
Unit 1: Retrieving, Restricting, and Sorting Data

Unit 2: Joins, Subqueries, and Set Operators

**Unit 3: DML and DDL**

- Lesson 10: Managing Tables Using DML Statements
- **Lesson 11: Introduction to Data Definition Language** — You are here!
- Lesson 12: Other Schema Objects

# HR Application Scenario



# Lesson Agenda

- Database objects
  - Naming rules
- `CREATE TABLE` statement
- Data types
- Overview of constraints: `NOT NULL`, `UNIQUE`, `PRIMARY KEY`, `FOREIGN KEY`, `CHECK` constraints
- Creating a table using a subquery
- `ALTER TABLE` statement
- `DROP TABLE` statement

# Database Objects

| Object | Description |
|---|---|
| Table | Is the basic unit of storage; composed of rows |
| View | Logically represents subsets of data from one or more tables |
| Sequence | Generates numeric values |
| Index | Improves the performance of some queries |
| Synonym | Gives alternative name to an object |

# Naming Rules for Tables and Columns

Ensure that the table names and column names:

- Begin with a letter
- Are 1–30 characters long
- Contain only A–Z, a–z, 0–9, _, $, and #
- Do *not* duplicate the name of another object owned by the same user
- Are *not* Oracle server–reserved words

# Lesson Agenda

- Database objects
  - Naming rules
- `CREATE TABLE` statement
- Data types
- Overview of constraints: `NOT NULL`, `UNIQUE`, `PRIMARY KEY`, `FOREIGN KEY`, `CHECK` constraints
- Creating a table using a subquery
- `ALTER TABLE` statement
- `DROP TABLE` statement

# CREATE TABLE Statement

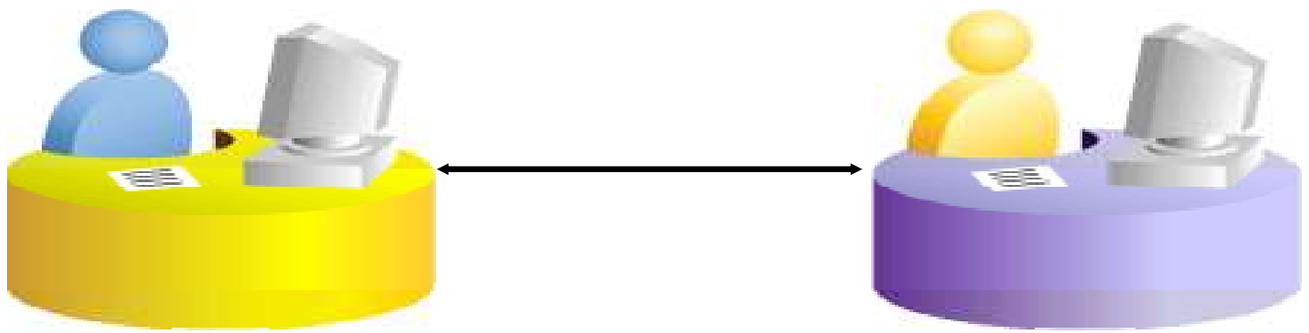
- You must have:
  - The CREATE TABLE privilege
  - A storage area

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

- You specify the:
  - 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.



**USERA**

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

**USERB**

```
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);
CREATE TABLE succeeded.
```

# Creating Tables

- Create the table:

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

```
table DEPT created.
```

- 

```
DESCRIBE dept
```

```
DESCRIBE dept
Name        Null Type
----------- ---- ------------
DEPTNO           NUMBER(2)
DNAME            VARCHAR2(14)
LOC              VARCHAR2(13)
CREATE_DATE      DATE
```

# Lesson Agenda

- Database objects
  - Naming rules
- `CREATE TABLE` statement
- Data types
- Overview of constraints: `NOT NULL`, `UNIQUE`, `PRIMARY KEY`, `FOREIGN KEY`, `CHECK` constraints
- Creating a table using a subquery
- `ALTER TABLE` statement
- `DROP TABLE` statement

# Data Types

| Data Type | Description |
|---|---|
| VARCHAR2(*size*) | Variable-length character data |
| CHAR(*size*) | Fixed-length character data |
| NUMBER(p, s) | Variable-length numeric data |
| DATE | Date and time values |
| LONG | Variable-length character data (up to 2 GB) |
| CLOB | Maximum size is (4 gigabytes - 1) * (DB_BLOCK_SIZE). |
| RAW and LONG RAW | Raw binary data |
| BLOB | Maximum size is (4 gigabytes - 1) * (DB_BLOCK_SIZE initialization parameter (8 TB to 128 TB)). |
| BFILE | Binary data stored in an external file (up to 4 GB) |
| ROWID | 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:

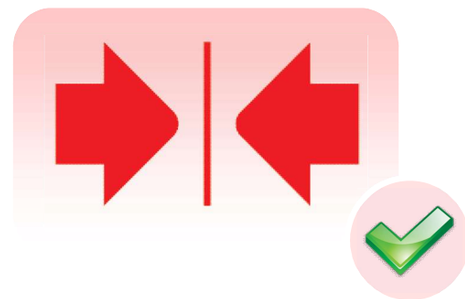
| Data Type | Description |
|---|---|
| `TIMESTAMP` | Date with fractional seconds |
| `INTERVAL YEAR TO MONTH` | Stored as an interval of years and months |
| `INTERVAL DAY TO SECOND` | Stored as an interval of days, hours, minutes, and seconds |

# Lesson Agenda

- Database objects
  - Naming rules
- `CREATE TABLE` statement
- Data types
- Overview of constraints: `NOT NULL`, `UNIQUE`, `PRIMARY KEY`, `FOREIGN KEY`, `CHECK` constraints
- Creating a table using a subquery
- `ALTER TABLE` statement
- `DROP TABLE` statement

# Including Constraints

- Constraints enforce rules at the table level.
- Constraints ensure consistency and integrity of the database.
- 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 time of table creation
  - After the creation of the table
- 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:

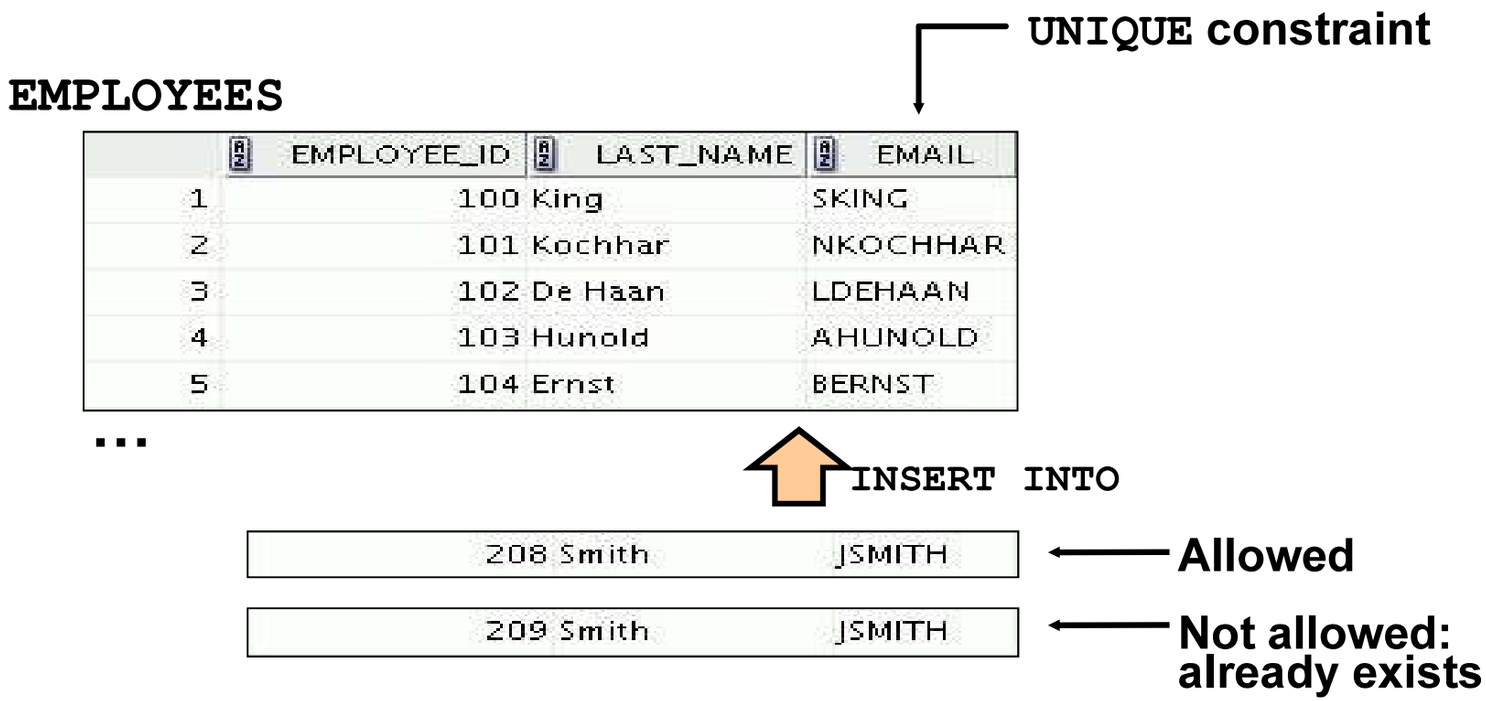
| | EMPLOYEE_ID | LAST_NAME | EMAIL | PHONE_NUMBER | HIRE_DATE | JOB_ID | SALARY | DEPARTMENT_ID |
|---|---|---|---|---|---|---|---|---|
| 1 | 178 | Grant | KGRANT | 011.44.1644.429263 | 24-MAY-99 | SA_REP | 7000 | (null) |
| 2 | 206 | Gietz | WGIETZ | 515.123.8181 | 07-JUN-94 | AC_ACCOUNT | 8300 | 110 |
| 3 | 205 | Higgins | SHIGGINS | 515.123.8080 | 07-JUN-94 | AC_MGR | 12000 | 110 |
| 4 | 100 | King | SKING | 515.123.4567 | 17-JUN-87 | AD_PRES | 24000 | 90 |
| 5 | 102 | De Haan | LDEHAAN | 515.123.4569 | 13-JAN-93 | AD_VP | 17000 | 90 |

...

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

**NOT NULL constraint**

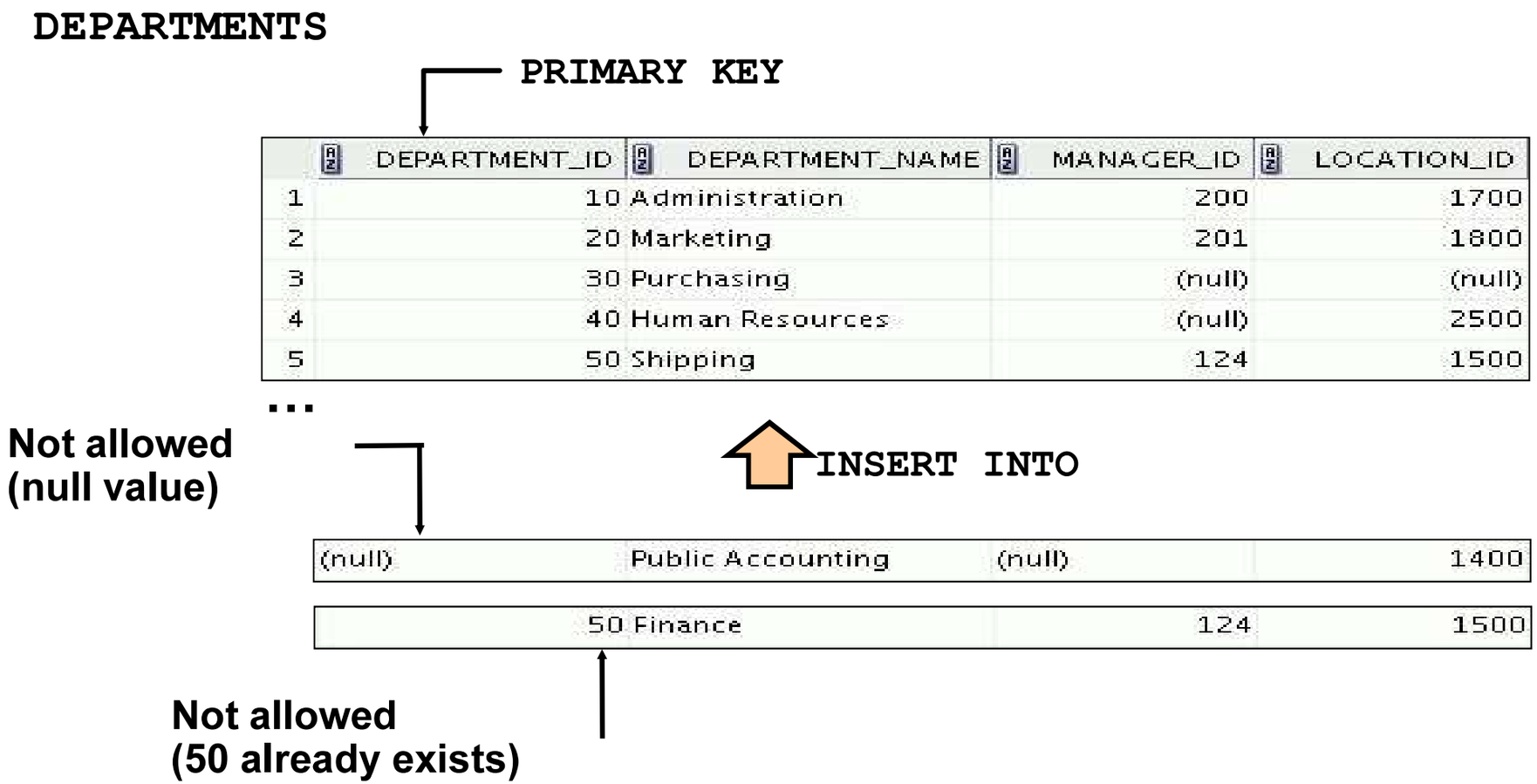
**Absence of NOT NULL constraint**
**(Any row can contain a null value for this column.)**


UNIQUE constraint
EMPLOYEES
EMPLOYEE_ID
LAST_NAME
EMAIL
1 100 King SKING
2 101 Kochhar NKOCHHAR
3 102 De Haan LDEHAAN
4 103 Hunold AHUNOLD
5 104 Ernst BERNST
...
INSERT INTO
208 Smith JSMITH
Allowed
209 Smith JSMITH
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));
```


DEPARTMENTS
PRIMARY KEY
DEPARTMENT_ID
DEPARTMENT_NAME
MANAGER_ID
LOCATION_ID
1 10 Administration 200 1700
2 20 Marketing 201 1800
3 30 Purchasing (null) (null)
4 40 Human Resources (null) 2500
5 50 Shipping 124 1500
...
Not allowed
(null value)
INSERT INTO
(null) Public Accounting (null) 1400
50 Finance 124 1500
Not allowed
(50 already exists)

# FOREIGN KEY Constraint

**DEPARTMENTS**

| | DEPARTMENT_ID | DEPARTMENT_NAME | MANAGER_ID | LOCATION_ID |
|---|---|---|---|---|
| 1 | 10 | Administration | 200 | 1700 |
| 2 | 20 | Marketing | 201 | 1800 |
| 3 | 30 | Purchasing | (null) | (null) |
| 4 | 40 | Human Resources | (null) | 2500 |
| 5 | 50 | Shipping | 124 | 1500 |

...

**PRIMARY KEY** →

**EMPLOYEES**

| | EMPLOYEE_ID | LAST_NAME | DEPARTMENT_ID |
|---|---|---|---|
| 1 | 100 | King | 90 |
| 2 | 101 | Kochhar | 90 |
| 3 | 102 | De Haan | 90 |
| 4 | 103 | Hunold | 30 |
| 5 | 104 | Ernst | 30 |

...

← **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
- Cannot reference columns from other tables

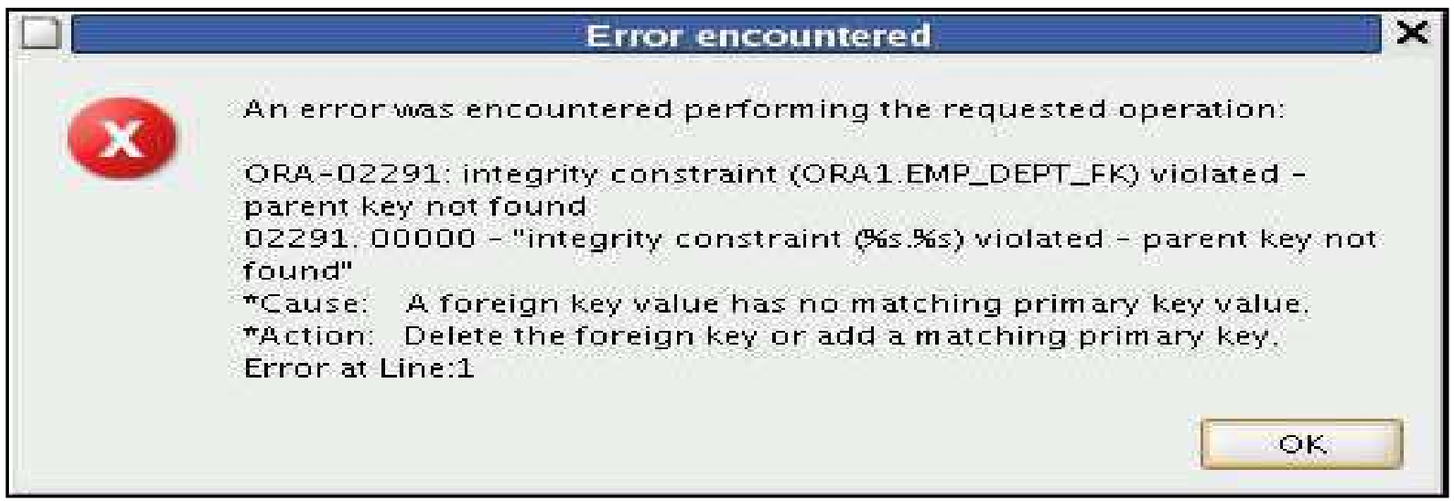
```
..., salary   NUMBER(2)
     CONSTRAINT emp_salary_min
             CHECK (salary > 0),...
```

# CREATE TABLE: Example

```
CREATE TABLE teach_emp (
        empno       NUMBER(5) PRIMARY KEY,
        ename       VARCHAR2(15) NOT NULL,
        job         VARCHAR2(10),
        mgr         NUMBER(5),
        hiredate    DATE DEFAULT (sysdate),
        photo       BLOB,
        sal         NUMBER(7,2),
        deptno      NUMBER(3) NOT NULL
                     CONSTRAINT admin_dept_fkey
                    REFERENCES
                      departments(department_id));
```

# Violating Constraints

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




- 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;
```

```
Error starting at line : 1 in command -
DELETE FROM departments
         WHERE department_id = 60
Error report -
SQL Error: ORA-02292: integrity constraint (TEACH_A.EMP_DEPT_FK) violated - child record found
02292. 00000 - "integrity constraint (%s.%s) violated - child record found"
*Cause:    attempted to delete a parent key value that had a foreign
           dependency.
*Action:   delete dependencies first then parent or disable constraint.
```

# Lesson Agenda

- Database objects
  - Naming rules
- Data types
- `CREATE TABLE` statement
- Overview of constraints: `NOT NULL`, `UNIQUE`, `PRIMARY KEY`, `FOREIGN KEY`, `CHECK` constraints
- Creating a table using a subquery
- `ALTER TABLE` statement
- `DROP TABLE` statement

# Creating a Table Using a Subquery

- Create a table and insert rows by combining the `CREATE TABLE` statement and

```
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 Using a Subquery

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

```
Table DEPT80 created.
```

```
DESCRIBE dept80
```

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

# Lesson Agenda

- Database objects
  - Naming rules
- Data types
- `CREATE TABLE` statement
- Overview of constraints: `NOT NULL`, `UNIQUE`, `PRIMARY KEY`, `FOREIGN KEY`, `CHECK` constraints
- Creating a table using a subquery
- `ALTER TABLE` statement
- `DROP TABLE` statement

# ALTER TABLE Statement

Use the ALTER TABLE statement to:

- Add a new column
- Modify an existing column definition
- Define a default value for the new column
- Drop a column
- Rename a column
- Change table to read-only status

# 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 [, column] …);
```

# Adding a Column

- You use the `ADD` clause to add columns:

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

```
table DEPT80 altered.
```

- The new column becomes the last column:

| | EMPLOYEE_ID | LAST_NAME | ANNSAL | HIRE_DATE | JOB_ID |
|---|---|---|---|---|---|
| 1 | 149 | Zlotkey | 10500 | 29-JAN-08 | (null) |
| 2 | 174 | Abel | 11000 | 11-MAY-04 | (null) |
| 3 | 176 | Taylor | 8600 | 24-MAR-06 | (null) |

# Modifying a Column

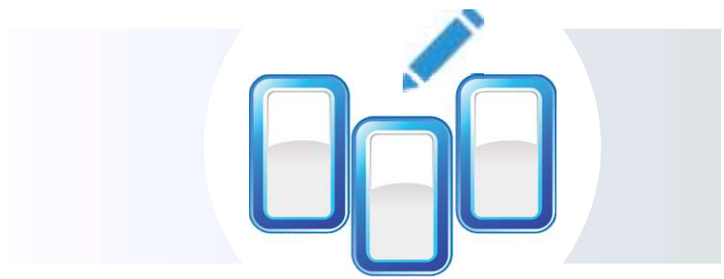
- You can change a column's data type, size, and default value.

```
ALTER TABLE dept80
MODIFY            (last_name VARCHAR2(30));
```

```
Table DEPT80 altered.
```

Size of the last_name column is modified.

- 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 that you no longer need from the table:

```
ALTER TABLE  dept80
DROP (job_id);
```

```
Table DEPT80 altered.
```

| | EMPLOYEE_ID | LAST_NAME | ANNSAL | HIRE_DATE |
|---|---|---|---|---|
| 1 | 149 | Zlotkey | 126000 | 29-JAN-16 |
| 2 | 174 | Abel | 132000 | 11-MAY-12 |
| 3 | 176 | Taylor | 103200 | 24-MAR-14 |
| 4 | 206 | Gietz | 99600 | 07-JUN-10 |

# 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.
- You can specify the ONLINE keyword to indicate that DML operations on the table will be allowed while marking the column or columns UNUSED.

```
ALTER TABLE        <table_name>
SET    UNUSED(<column_name> [ , <column_name>]);
OR
ALTER TABLE  <table_name>
SET    UNUSED COLUMN <column_name> [,<column_name>];
```

```
ALTER TABLE <table_name>
DROP  UNUSED COLUMNS;
```

# Read-Only Tables

You can use the `ALTER TABLE` syntax to:

- Put a table in read-only mode, which prevents DDL or DML changes during table maintenance
- Put the table back into read/write mode

```
ALTER TABLE employees READ ONLY;

-- perform table maintenance and then
-- return table back to read/write mode

ALTER TABLE employees READ WRITE;
```

# Dropping a Table

- Moves a table to the recycle bin
- Removes the table and all its data entirely if the `PURGE` clause is specified
- Invalidates dependent objects and removes object privileges on the table

```
DROP TABLE dept80;
```

```
Table DEPT80 dropped.
```

# Quiz

Identify three actions that you perform by using constraints.

a. Enforce rules on the data in a table whenever a row is inserted, updated, or deleted.

b. Prevent the dropping of a table.

c. Prevent the creation of a table.

d. Prevent the creation of data in a table.

# Summary

In this lesson, you should have learned how to use the `CREATE TABLE`, `ALTER TABLE`, and `DROP TABLE` statement to create a table, modify a table and columns, 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
- Explain how constraints are created at the time of table creation

# Practice 9: Overview

This practice covers the following topics:

- Creating new tables
- Creating a new table by using the `CREATE TABLE AS` syntax
- Verifying that tables exist
- Altering tables
- Adding columns
- Dropping columns
- Setting a table to read-only status
- Dropping tables