# Oracle 11g - SQL

## Using DDL Statements to Create and Manage Tables

# Objectives

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

- Categorize SQL database objects
- Naming Conventions & List the data types available in SQL
- Review on `CREATE TABLE` privilege to others Users
- Applying Database Level Object privilege
- Impact Explained: Violating, The Principle of Least Privilege...
- Example on Database level Object permission
- Create a simple table with Data types
- Create  Table with simple data type along with constraints
- Creating a Table by Using a Sub-query
- Alter Command to manage table Object

# Database Objects

| Object | Description |
|---|---|
| Table | 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 names to objects |

# 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

# Data Types

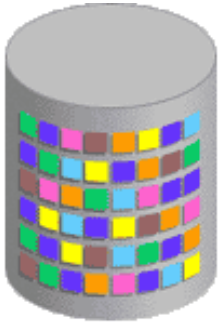
| Data Type | Description |
|---|---|
| VARCHAR2(*size*) | Variable-length character data |
| CHAR(*size*) | Fixed-length character data |
| NUMBER(*p*,*s*) | Variable-length numeric data |
| DATE | Date and time values |
| LONG | Variable-length character data (up to 2 GB) |
| CLOB | Character data (up to 4 GB) |
| RAW and LONG RAW | Raw binary data |
| BLOB | Binary data (up to 4 GB) |
| 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 |

# 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



# 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);
CREATE TABLE succeeded.
```

- Confirm table creation.

```
DESCRIBE dept
```

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

4 rows selected
```

# 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 with 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 |
|---|---|---|---|---|---|---|---|---|
| 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 |

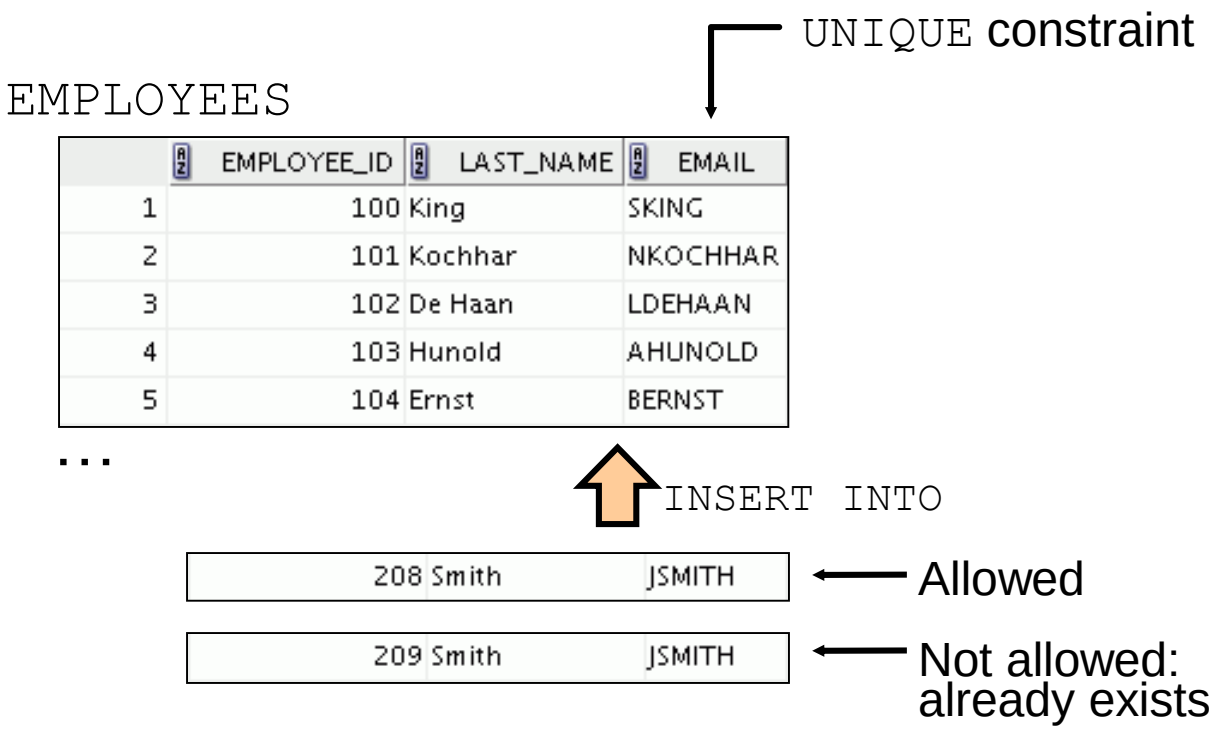
...

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

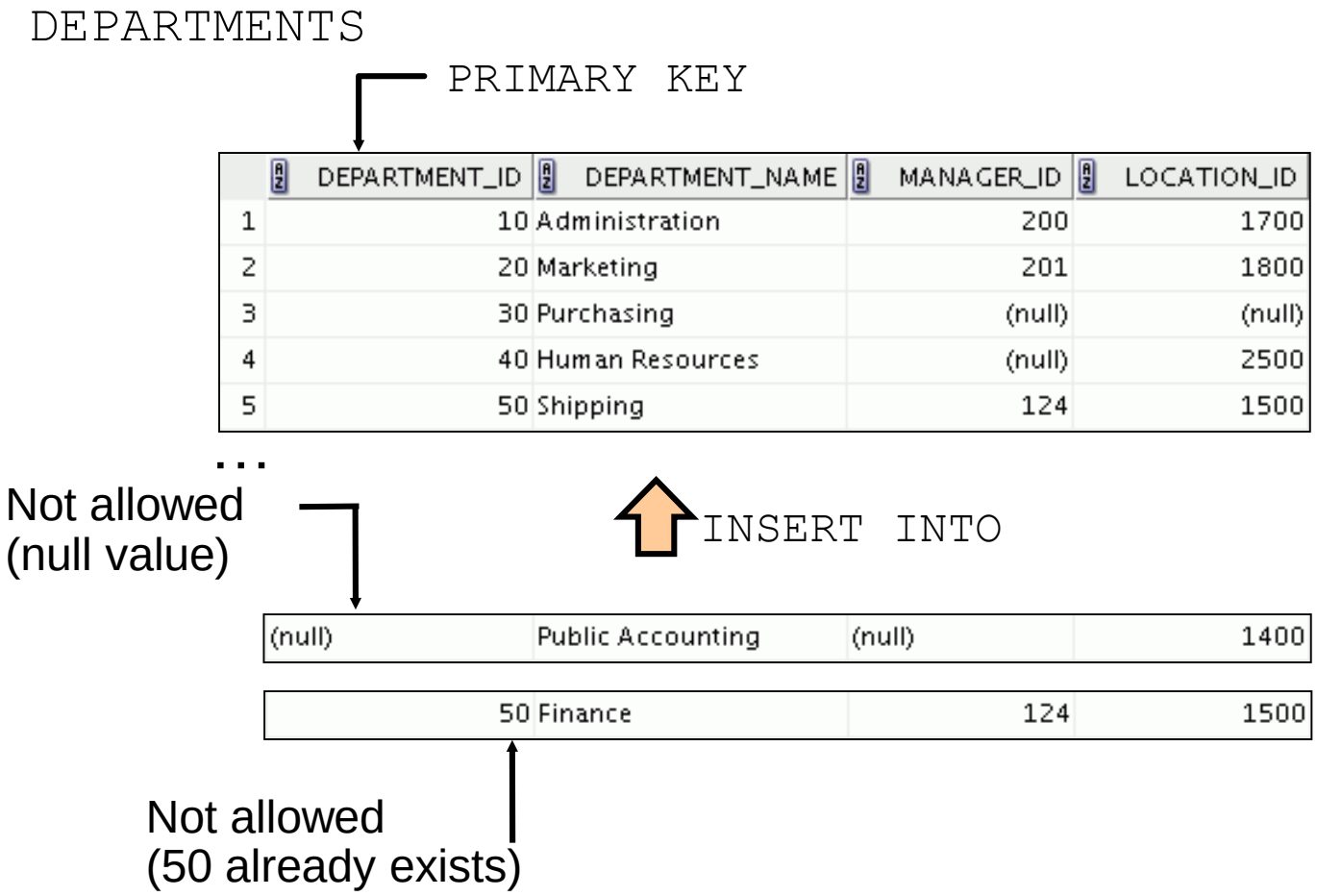


# UNIQUE Constraint

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



# 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
- The following expressions are not allowed:
  - References to CURRVAL, NEXTVAL, LEVEL, and ROWNUM pseudocolumns
  - Calls to SYSDATE, UID, USER, and USERENV functions

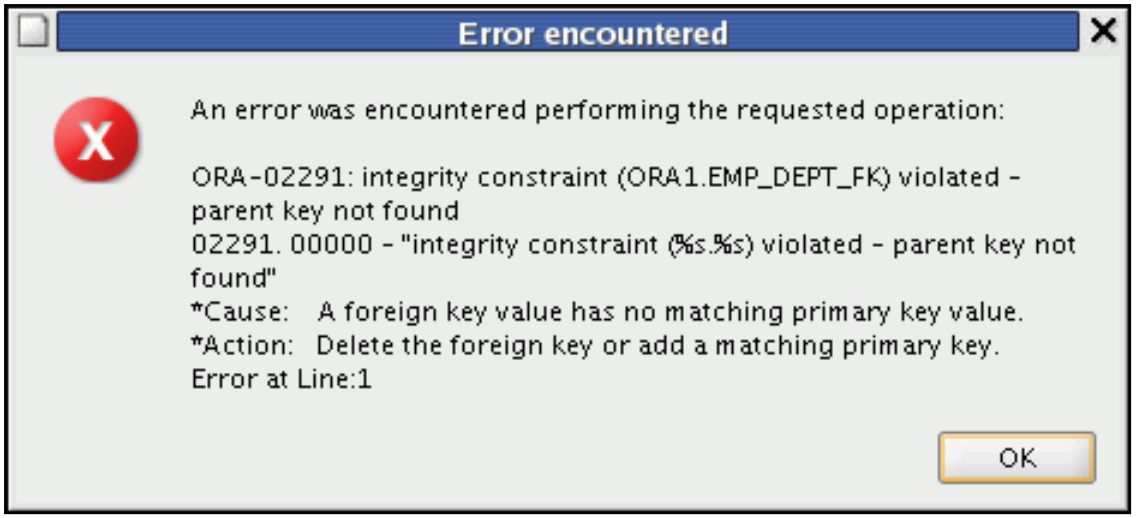
```
..., salary   NUMBER(2)
     CONSTRAINT emp_salary_min
            CHECK (salary > 0),...
```

# 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;
```

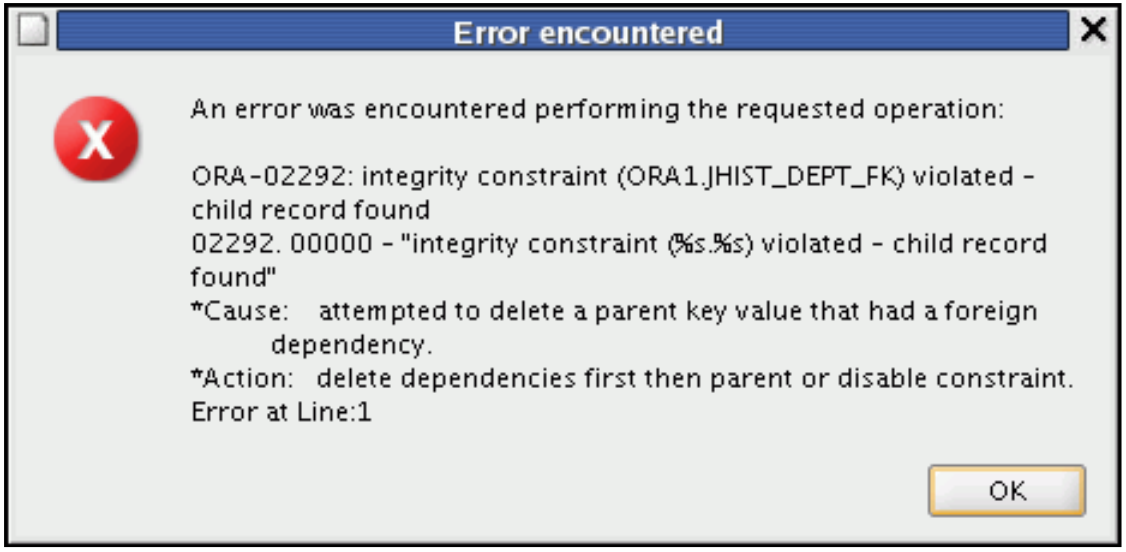



Department 55 does not exist.

# 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;
```



# 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;
CREATE TABLE Succeeded.
```

```
DESCRIBE dept80
```

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

4 rows selected
```

# ALTER TABLE Statement

Use the ***ALTER TABLE*** statement to:

- **Add** a New column

*ALTER TABLE table_name ADD column_name column-definition;*

- **Modify** an Existing column

*ALTER TABLE table_name MODIFY column_name column_type;*

- **Rename** the existing Table

*ALTER TABLE table_name RENAME TO new_table_name;*

- **Drop** a column

*ALTER TABLE table_name DROP COLUMN column_name;*

# 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;
DROP TABLE succeeded.
```