# DATABASE DESIGN & IMPLEMENTATION

ICT Skills

# Objectives

- INSERT statement using the values clause
- INSERT special values, null values and date values
- INSERT to copy rows from one table to another

# Purpose

- Making changes to the database
- Databases are dynamic
- Constantly inserting updating and deleting data.

# Copy Tables before inserting

- To keep your schema tables in their original state, you will make a copy of each table before completing the practice activities.
- You will use the copy of the table that you create not the original to insert data.
- If you make changes in error then you can use the original table to restore the copy.
- You should name each copied table: copy_tablename.
- The copied table does not inherit the primary key to foreign key integrity rules of the original table.
- The column data types are inherited in the copied table.

# Syntax to create a copy of a table

- Create table syntax:

```
CREATE TABLE copy_tablename
   AS (SELECT * FROM tablename);
```

- For example:

```
CREATE TABLE copy_employees
AS (SELECT * FROM employees);
```

```
CREATE TABLE copy_departments
AS (SELECT * FROM departments);
```

# Syntax to create a copy of a table

- To verify and view the copy of the table you DESCRIBE it.

```
DESCRIBE copy_employees;
```

```
DESCRIBE copy_departments;
```

# INSERT

- The INSERT statement is used to add a new row to a table.
- The statement requires three values:
  - *The name of the table*
  - *The names of the columns in the table to populate*
  - *Corresponding values for each column*
- We want to INSERT the data below in a copy of the department table:

| DEPARTMENT_ID | DEPARTMENT_NAME | MANAGER_ID | LOCATION_ID |
|---|---|---|---|
| 200 | Human Resources | 205 | 1500 |

# INSERT

- The syntax uses INSERT to add a new department to the copy_department table.
- This statement explicitly lists each column as it appears in the table.
- The values for each column are listed in the same order.
- Note that the number values are not enclosed in quotes.

```
INSERT INTO copy_departments
   (department_id, department_name, manager_id, location_id)
VALUES
   (200,'Human Resources', 205, 1500);
```

| DEPARTMENT_ID | DEPARTMENT_NAME | MANAGER_ID | LOCATION_ID |
|---|---|---|---|
| 200 | Human Resources | 205 | 1500 |

# INSERT

- Another way to insert values is to implicitly add them by omitting the column names.
- One precaution: the values for each column must match exactly the default order in which they appear in the table and a value must be provided for each column

```
INSERT INTO copy_departments
VALUES
  (210,'Estate Management', 102, 1700);
```

| DEPARTMENT_ID | DEPARTMENT_NAME | MANAGER_ID | LOCATION_ID |
|---|---|---|---|
| 210 | Estate Management | 102 | 1700 |

# Check the table first

- Before inserting data into a table, you must check several table details.
- The DESCRIBE tablename statement will return a description of the table strucutre.

| Column Name | Data Type | Nullable | Default | Primary Key |
|---|---|---|---|---|
| DEPARTMENT_ID | NUMBER(4,0) | Yes | - | - |
| DEPARTMENT_NAME | VARCHAR2(30) | No | - | - |
| MANAGER_ID | NUMBER(6,0) | Yes | - | - |
| LOCATION_ID | NUMBER(4,0) | Yes | - | - |

- Number specifies the precision and scale. Precision is the total number of digits, scale is the number of digits to the right of the decimal place.

# Inserting rows with NULL values

- The INSERT statement need not specify every column.
- If every column that requires a value is assigned a value then the insert will work.
- An implicit insert will automatically insert a null value in columns that allow nulls.
- To explicitly add a null value to a column that allows nulls, use the NULL keyword in the VALUES list.

```
INSERT INTO copy_employees
  (employee_id, first_name, last_name, phone_number, hire_date,
   job_id, salary)
VALUES
  (302,'Grigorz','Polanski', '8586667641', '15/Jun/2015',
   'IT_PROG',4200);
```

```
ORA-01400: cannot insert NULL into
("US_A009EMEA815_PLSQL_T01"."COPY_EMPLOYEES"."EMAIL")
```

# Inserting empty strings

- To specify empty strings use empty single quotation marks for missing data.

```
INSERT INTO copy_employees
  (employee_id, first_name, last_name, email, phone_number,
   hire_date, job_id, salary)
VALUES
  (302,'Grigorz','Polanski', 'gpolanski', '', '15/Jun/2015',
   'IT_PROG',4200);
```

# Inserting special values

- Special values such as SYSDATE and USER can be entered in the VALUES list of an INSERT statement.
- SYSDATE will put the current date and time in a column
- USER will insert the current session's username, which in Oracle Application Express is OAE_PUBLIC_USER

```
INSERT INTO copy_employees
  (employee_id, first_name, last_name, email, phone_number, hire_date,
   job_id, salary)
VALUES
  (304,'Test',USER, 't_user', 4159982010, SYSDATE, 'ST_CLERK',2500);
```

# Inserting date values

- The default format model for date data types is DD-Mon-YYYY.
- If we want to INSERT a row with a non-default format for a date column we must use the TO_DATE function to convert the date value ( a character string) to a date.

```
INSERT INTO copy_employees
  (employee_id, first_name, last_name, email, phone_number, hire_date,
   job_id, salary)
VALUES
   (301,'Katie','Hernandez', 'khernandez','8586667641',
    TO_DATE('July 8, 2015', 'Month fmdd, yyyy'), 'MK_REP',4200);
```

# Using a subquery to copy rows

- Each INSERT statement we have seen so far adds only one row to the table.
- But suppose we want to copy 100 rows from one table to another.
- SQL allows a subquery within an INSERT statement to do this.
- All the rows are inserted in the table.
- We don’t need a VALUES clause.

```
INSERT INTO sales_reps
   SELECT *
   FROM   employees;
```

# Using a subquery to copy rows

- The number of columns and their data types in the column list of the INSERT clause must match the number of columns and their data types in the subquery.
- No parenthesis around the subquery.