GIFT School of Engineering and Applied Sciences

Fall 2022

CS-244: Database Systems-Lab

Lab-05 Manual

Manipulating Data Using DML

Introduction to Lab

In this lab, you will learn how to insert rows into a table, update existing rows in a table, and delete existing rows from a table.

The main topics of this lab include:

1. Data Manipulation Language (DML)
2. The INSERT Statement Syntax
3. Inserting New Rows
4. Inserting Rows with Null Values
5. Inserting Special Values
6. Creating a Script
7. Changing Data in a Table (The UPDATE Statement)
8. Updating Rows in a Table
9. Updating Rows: Integrity Constraint Error
10. Removing a Row from a Table (The DELETE Statement)
11. Deleting Rows from a Table
12. Deleting Rows Based on Another Table
13. Deleting Rows: Integrity Constraint Error
14. The TRUNCATE Statement

Objectives of this Lab

At the end of this lab, students should be able to:

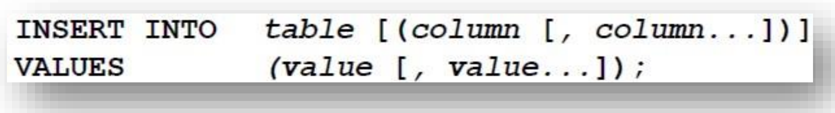
1. Describe each DML statement
2. Insert rows into a table
3. Update rows in a table
4. Delete rows from a table
5. Merge rows in a table

1. Data Manipulation Language (DML)

Data manipulation language (DML) is a core part of SQL. When you want to add, update, or delete data in the database, you execute a DML statement.

2. The INSERT Statement Syntax

Add new rows to a table by using the INSERT statement.



```
INSERT INTO   table [(column [, column...])]
VALUES        (value [, value...]);
```

In the syntax:

table	**is the name of the table**
column	is the name of the column in the table to populate
value	is the corresponding value for the column

Note: This statement with the VALUES clause adds only one row at a time to a table.

3. Inserting New Rows

```
INSERT INTO departments(department_id,
department_name, manager_id, location_id) VALUES
(70, 'Public Relations', 100, 1700);
```

Because you can insert a new row that contains values for each column, the column list is not required in the INSERT clause. However, if you do not use the column list, the values must be listed according to the default order of the columns in the table, and a value must be provided for each column.

Note: For clarity, use the column list in the INSERT clause.

Enclose character and date values within single quotation marks; it is not recommended to enclose numeric values within single quotation marks.

4. Inserting Rows with Null Values

Methods for Inserting Null Values

Method	**Description**
Implicit	Omit the column from the column list.

Explicit	Specify the `NULL` keyword in the `VALUES` list, specify the empty string ('') in the `VALUES` list for character strings and dates.

The Oracle Server automatically enforces all data types, data ranges, and data integrity constraints. Any column that is not listed explicitly obtains a null value in the new row.

Examples:

Implicit method: Omit the column from the column list.

```
INSERT INTO departments (department_id, department_name)
VALUES (30, 'Purchasing');
```

Explicit method: Specify the NULL keyword in the VALUES clause.

```
INSERT INTO departments
VALUES (100, 'Finance', NULL, NULL);
```

Common errors: that can occur during user input:

- Mandatory value missing for a NOT NULL column
- Duplicate value violates uniqueness constraint
- Foreign key constraint violated
- CHECK constraint violated
- Data type mismatch
- Value too wide to fit in column

5. Inserting Special Values

You can use functions to enter special values in your table.

```
INSERT INTO employees (employee_id,
first_name, last_name, email,
phone_number, hire_date, job_id,
salary, commission_pct, manager_id,
department_id) VALUES (113, 'Louis',
'Popp', 'LPOPP', '515.124.4567',
SYSDATE, 'AC_ACCOUNT', 6900, NULL,
205, 100);
```

The above example records information for employee Popp in the EMPLOYEES table. It supplies the current date and time in the HIRE_DATE column. It uses the SYSDATE function for current date and time.

Confirming Additions to the Table

```
SELECT employee_id, last_name, job_id, hire_date,
commission_pct FROM employees WHERE employee_id =
113;
```

6. Creating a Script

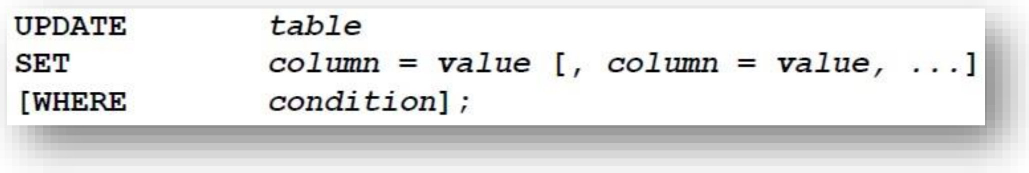
You can save commands with substitution variables to a file and execute the commands in the file. The slide example records information for a department in the DEPARTMENTS table.

```
INSERT INTO departments
(department_id, department_name, location_id)
VALUES (&department_id, '&department_name',&location);
```

Run the script file and you are prompted for input for each of the & substitution variables. The values that you input are then substituted into the statement. This enables you to run the same script file over and over but supply a different set of values each time you run it.

7. Changing Data in a Table (The UPDATE Statement)

You can modify existing rows by using the UPDATE statement.



```
UPDATE        table
SET           column = value [, column = value, ...]
[WHERE        condition];
```

In the syntax:	
table	is the name of the table
column	is the name of the column in the table to populate
value	is the corresponding value or subquery for the column
condition	identifies the rows to be updated and is composed of column names expressions, constants, subqueries, and comparison operators

Confirm the update operation by querying the table to display the updated rows.

Note: In general, use the primary key to identify a single row. Using other columns can unexpectedly cause several rows to be updated. For example, identifying a single row in the **EMPLOYEES** table by name is dangerous, because more than one employee may have the same name.

8. Updating Rows in a Table

The UPDATE statement modifies specific rows if the WHERE clause is specified.

Note: If you omit the WHERE clause, all the rows in the table are modified.

Example:

```
UPDATE employees
SET department_id = 70
WHERE employee_id = 113;
```

Note: All rows in the table are modified if you omit the WHERE clause:

```
UPDATE copy_emp
SET department_id = 110;
```

9. Updating Rows: Integrity Constraint Error

If you attempt to update a record with a value that is tied to an integrity constraint, an error is returned.

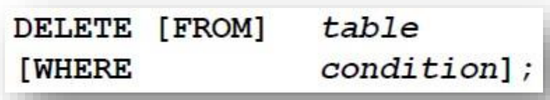
In this example, department number 55 does not exist in the parent table, DEPARTMENTS, and so you receive the parent key violation error.

```
UPDATE copy_emp SET department_id = 55 WHERE department_id = 90;
```

Note: Integrity constraints ensure that the data adheres to a predefined set of rules. A subsequent lab covers integrity constraints in greater depth.

10. Removing a Row from a Table (The DELETE Statement)

You can remove existing rows by using the DELETE statement.



```
DELETE [FROM]   table
[WHERE          condition];
```

In the syntax:	
table	is the table name
condition	identifies the rows to be deleted and is composed of column names, expressions, constants, subqueries, and comparison operators

Note: If no rows are deleted, a message "`0 rows deleted.`" is returned:

11. Deleting Rows from a Table

You can delete specific rows by specifying the **WHERE** clause in the **DELETE** statement.

```
DELETE FROM departments WHERE department_name = 'Finance';
```

The above example deletes the Finance department from the DEPARTMENTS table. You can confirm the delete operation by displaying the deleted rows using the SELECT statement.

```
SELECT * FROM departments WHERE department_name = 'Finance';
```

If you omit the WHERE clause, all rows in the table are deleted. The second example below deletes all the rows from the COPY_EMP table, because no WHERE clause has been specified.

```
DELETE FROM copy_emp;
```

Example:

Remove rows identified in the WHERE clause.

```
DELETE FROM employees WHERE employee_id = 114;
```

```
DELETE FROM departments WHERE department_id IN (30, 40);
```

12. Deleting Rows Based on Another Table

You can use subqueries to delete rows from a table based on values from another table.

The example below deletes all the employees who are in a department where the department name contains the string ‘Public’. The subquery searches the DEPARTMENTS table to find the department number based on the department name containing the string ‘Public’. The subquery then feeds the department number to the main query, which deletes rows of data from the COPY_EMP table based on this department number.

```
DELETE FROM copy_emp WHERE department_id = (SELECT department_id FROM
departments WHERE department_name LIKE '%Public%');
```

13. Deleting Rows: Integrity Constraint Error

If you attempt to delete a record with a value that is tied to an integrity constraint, an error is returned.

```
DELETE FROM departments WHERE department_id = 90;
```

The above example tries to delete department number 40 from the DEPARTMENTS table, but it results in an error because department number is used as a foreign key in the COPY_EMP table. If the parent record that you attempt to delete has child records, then you receive the child record found violation error.

14. The TRUNCATE Statement

A more efficient method of emptying a table is with the TRUNCATE statement.

```
TRUNCATE TABLE table_name;
```

Example:

```
TRUNCATE TABLE copy_emp;
```

Note: No rollback is possible after executing the TRUNCATE statement!

Removing rows with the TRUNCATE statement is faster than removing them with the DELETE statement for the following reasons:

- The TRUNCATE statement is a data definition language (DDL) statement and generates no rollback information. Rollback information is covered later in this lab.
- If the table is the parent of a referential integrity constraint, you cannot truncate the table. You need to disable the constraint before issuing the TRUNCATE statement. Disabling constraints is covered in a subsequent lab.

The End