Lecture Notes – 3 InsertingData-DML

Section 1: Lecture Summary

This lecture covers the fundamentals of inserting data into MySQL database tables using DML (Data Manipulation Language). The focus is on the **INSERT** statement, which is one of three DML commands (INSERT, UPDATE, DELETE). The lecture uses the companyDB database and its Departments and Employees tables to demonstrate how data insertion works through practical examples from a downloaded SQL setup script.

Section 2: Key Concepts and Explanations

INSERT Statement Syntax

The basic syntax for inserting data is:

```
INSERT INTO table_name (column1, column2, column3)
VALUES (value1, value2, value3);
```

Column Order Flexibility

When specifying columns explicitly, you can list them in any order, and the values must follow the same order. For example:

```
INSERT INTO Departments (Location, DeptName, DeptID)
VALUES ('Mumbai', 'Human Resource', 1);
```

This is valid even though the column order differs from the table definition.

Omitting Column Names

If you insert values in the exact order that columns are defined in the table, you can omit the column names:

```
INSERT INTO Departments
VALUES (1, 'Human Resource', 'Mumbai');
```

Multiple Row Insertion

You can insert multiple rows in a single statement by providing multiple value sets separated by commas:

```
INSERT INTO Departments (DeptID, DeptName, Location)
VALUES
(1, 'Human Resource', 'Mumbai'),
(2, 'IT', 'Bangalore'),
(3, 'Finance', 'Delhi'),
(4, 'Operations', 'Chennai');
```

Handling Missing Data with NULL

When data for a specific column is unavailable, you can use the **NULL** keyword as a placeholder:

```
INSERT INTO Employees (EmpID, FirstName, LastName, JobTitle, DeptID,
HireDate, Salary)
VALUES (101, 'John', 'Smith', 'Manager', 2, NULL, 78000);
```

However, if a column is defined as NOT NULL, you cannot insert NULL values and must provide actual data.

Section 3: Example Code and Use Cases

Example 1: Basic Single Row Insertion

```
INSERT INTO Departments (DeptID, DeptName, Location)
VALUES (1, 'Human Resource', 'Mumbai');
```

Example 2: Insertion with Column Order Changed

```
INSERT INTO Departments (Location, DeptName, DeptID)
VALUES ('Delhi', 'Finance', 3);
```

Example 3: Insertion Without Column Names (Same Table Order)

```
INSERT INTO Departments
VALUES (2, 'IT', 'Bangalore');
```

Example 4: Multiple Rows Insertion

```
INSERT INTO Departments (DeptID, DeptName, Location)
VALUES
(1, 'Human Resource', 'Mumbai'),
(2, 'IT', 'Bangalore'),
(3, 'Finance', 'Delhi'),
(4, 'Operations', 'Chennai');
```

Example 5: Employee Data with NULL Values

```
INSERT INTO Employees (EmpID, FirstName, LastName, JobTitle, DeptID,
HireDate, Salary)
VALUES (101, 'John', 'Smith', 'Manager', 2, NULL, 78000);
```

Example 6: Employee Data with All Values Provided

```
INSERT INTO Employees (EmpID, FirstName, LastName, JobTitle, DeptID,
HireDate, Salary)
VALUES (102, 'Sarah', 'Johnson', 'Developer', 2, '2023-01-15', 85000);
```

Section 4: Key Takeaways

Column Specification: Always specify column names if inserting values in a different order than the table definition, or omit them if following the exact table column sequence.

NULL Usage: Use NULL for missing data only if the column allows it (NOT NULL constraint not applied). Missing required data prevents insertion.

Batch Insertion: Multiple rows can be inserted efficiently in a single statement by using multiple value sets with comma separation.

Data Integrity: The INSERT statement is the foundational DML command for populating database tables, and understanding its syntax is essential before progressing to SELECT queries and other data manipulation operations.