Lecture Notes – What is DDL?

Section 1: Lecture Summary

DDL stands for **data definition language**, a type of SQL used for creating, altering, renaming, truncating, and dropping tables. The lecture covers the **CREATE TABLE** syntax in detail, including column definitions with data types, sizes, and constraints like **AUTO_INCREMENT**, **NOT NULL**, **UNIQUE**, **CHECK**, **DEFAULT**, **PRIMARY KEY**, and **FOREIGN KEY**. An example table named **employees2** is defined using various data types (numeric like **INT**, **DECIMAL**; string like **VARCHAR**; date like **DATE**; boolean like **BOOL**) and constraints, referencing the existing **Departments** table via **DeptID** as a foreign key.

Section 2: Key Concepts and Explanations

CREATE TABLE syntax starts with `CREATE TABLE table_name (column1 datatype(size) constraint, column2 datatype constraint, ...);` followed by table-level constraints like **PRIMARY KEY** and **FOREIGN KEY**.

Data types include:

- Numeric: **INT** (fixed size, e.g., 4 bytes), **SMALLINT** or **BIGINT** for varying integer sizes, **DECIMAL(M,D)** for precise decimals (M total digits, D after decimal), **FLOAT** or **DOUBLE** for approximate decimals.

- String: **VARCHAR(n)** for variable-length text up to n characters, **TEXT**, **MEDIUMTEXT**, **LONGTEXT** for longer text.

- Date/Time: **DATE** for dates, **TIME** for times, **DATETIME**, **TIMESTAMP** for combined.

Constraints enforce rules:

- **AUTO_INCREMENT** for automatic sequential values (e.g., on **EmpID**).

- **NOT NULL** prevents empty values.

- **UNIQUE** ensures no duplicates (e.g., on email).

- **CHECK (condition)** validates data (e.g., `CHECK (salary > 0)`).

- **DEFAULT value** sets automatic values (e.g., `DEFAULT CURRENT_DATE` on **HireDate**, `DEFAULT TRUE` on active flag).

- **PRIMARY KEY** uniquely identifies rows.

- **FOREIGN KEY (column) REFERENCES other_table(other_column)** links to another table's primary key.

Table design requires pre-deciding columns, data types, sizes, and constraints during database design.

Section 3: Example Code and Use Cases

Using **companyDB** schema, here is the **CREATE TABLE** query for **employees2** as defined, adapted to match existing **Employees** and **Departments** structures:

```
CREATE TABLE employees2 (
    EmpID INT AUTO_INCREMENT PRIMARY KEY,
    Name VARCHAR(50) NOT NULL,
    Email VARCHAR(100) UNIQUE,
    Salary DECIMAL(8,2) CHECK (Salary > 0),
    HireDate DATE DEFAULT CURRENT_DATE,
    IsActive BOOL DEFAULT TRUE,
    DeptID INT,
    FOREIGN KEY (DeptID) REFERENCES Departments(DeptID)
);
```

This creates a table with **EmpID** as auto-incrementing primary key, **DeptID** as foreign key to **Departments**, and constraints ensuring data integrity similar to the **Employees** table.

Section 4: Key Takeaways

- Use **CREATE TABLE** with columns defined as `column datatype(size) [constraints]`, followed by table-level **PRIMARY KEY** and **FOREIGN KEY**.

- Select data types based on data needs: **INT** for IDs, **VARCHAR** for names/emails, **DECIMAL** for salaries, **DATE** for dates.

- Apply constraints like **AUTO_INCREMENT**, **NOT NULL**, **UNIQUE**, **CHECK**, and **DEFAULT** at column level; keys at table level.

- Design tables carefully upfront, matching existing schemas like **companyDB**'s **Employees** and **Departments**.