Lecture Notes – Database Terminology

Section 1: Lecture Summary

The lecture introduces fundamental database terminology using an employees table example with columns like employee id, first name, last name, job title, department id, and salary. It explains that a table is a **relation**, rows are **records** (also called tuples), columns are **fields** (also called attributes), **primary key** uniquely identifies rows and cannot be null, **foreign key** establishes relationships between tables by referencing a primary key (like department id linking employees to departments), and this creates **constraints** for referential integrity ensuring only valid values are used.

Section 2: Key Concepts and Explanations

- **Relation**: A table, representing interrelated data in rows and columns.

- **Record (Row/Tuple)**: A single row containing information for one entity, such as one employee.

- **Field (Column/Attribute)**: A column holding specific data type, like employee id or salary.

- **Primary key**: A unique, non-null column (e.g., **EmpID** in Employees) used to distinctly identify each row.

- **Foreign key**: A column in one table (e.g., **DeptID** in Employees) that references the primary key in another table (e.g., **DeptID** in Departments), forming relationships.

- **Constraint**: Rule enforced by foreign key for referential integrity, ensuring foreign key values exist in the referenced primary key (e.g., Employees **DeptID** must match a Departments **DeptID**).

Relationships connect tables, making the database a collection of interrelated data.

Section 3: Example Code and Use Cases

Using companyDB schemas:

```
-- View Employees table (relation) with records and fields
SELECT * FROM Employees;
-- Shows rows as records (e.g., one per EmpID), columns as fields (e.g.,
```

```
FirstName, DeptID).

-- Primary key example: EmpID uniquely identifies each employee
SELECT EmpID, FirstName, LastName
FROM Employees
WHERE EmpID = 1;  -- Uses primary key for distinct lookup

-- Foreign key relationship: Join Employees and Departments via DeptID
SELECT e.FirstName, e.LastName, e.JobTitle, d.DeptName
FROM Employees e
JOIN Departments d ON e.DeptID = d.DeptID;  -- e.DeptID (foreign key)
references d.DeptID (primary key)
-- Enforces constraint: only valid DeptID values from Departments.

-- Projects related via foreign keys
SELECT e.FirstName, p.ProjectName
FROM Employees e
JOIN EmployeeProjects ep ON e.EmpID = ep.EmpID
JOIN Projects p ON ep.ProjectID = p.ProjectID AND e.DeptID = p.DeptID;
```

Section 4: Key Takeaways

- Tables are **relations** with **rows** (**records/tuples**) and **columns** (**fields/attributes**).

- **Primary key** ensures uniqueness and non-null for row identification (e.g., **EmpID**, **DeptID**).

- **Foreign key** links tables for relationships (e.g., **DeptID** in Employees to Departments).

- **Constraints** maintain referential integrity between related tables.