Lecture Notes – Database Design -- CompanyDB

Section 1: Lecture Summary

The lecture covers designing the **companyDB** database for a company with employees, departments, and projects. It identifies entities (departments, employees, projects), their attributes, and relationships: one-to-many between departments and employees, one-to-many between departments and projects, and many-to-many between employees and projects resolved via an EmployeeProjects table. The design uses primary and foreign keys to establish relationships, resulting in four tables with sample data for practicing SQL queries.

Section 2: Key Concepts and Explanations

Database design starts by identifying **entities** (departments, employees, projects) and a relationship entity (EmployeeProjects). Each entity has attributes: Departments (DeptID, DeptName, Location), Employees (EmpID, FirstName, LastName, JobTitle, DeptID, HireDate, Salary), Projects (ProjectID, ProjectName, DeptID), EmployeeProjects (EmpID, ProjectID). Relationships are determined by cardinality: one department has many employees (DeptID as foreign key in Employees), one department handles many projects (DeptID as foreign key in Projects), one employee works on many projects and one project involves many employees (many-to-many, resolved by EmployeeProjects with EmpID and ProjectID as composite primary key and foreign keys). Foreign keys ensure referential integrity, like DeptID in Employees matching existing DeptID values in Departments. The design provides a small dataset (4 departments, 20 employees) for SQL practice before scaling to larger data.

Section 3: Example Code and Use Cases

```
USE companyDB;


-- Create tables with relationships

CREATE TABLE Departments (

  DeptID INT PRIMARY KEY,

  DeptName VARCHAR(50),

  Location VARCHAR(50)
```

```
);

CREATE TABLE Employees (
  EmpID INT PRIMARY KEY,
  FirstName VARCHAR(50),
  LastName VARCHAR(50),
  JobTitle VARCHAR(50),
  DeptID INT,
  HireDate DATE,
  Salary DECIMAL(10,2),
  FOREIGN KEY (DeptID) REFERENCES Departments(DeptID)
);

CREATE TABLE Projects (
  ProjectID INT PRIMARY KEY,
  ProjectName VARCHAR(100),
  DeptID INT,
  FOREIGN KEY (DeptID) REFERENCES Departments(DeptID)
);

CREATE TABLE EmployeeProjects (
  EmpID INT,
  ProjectID INT,
  PRIMARY KEY (EmpID, ProjectID),
  FOREIGN KEY (EmpID) REFERENCES Employees(EmpID),
  FOREIGN KEY (ProjectID) REFERENCES Projects(ProjectID)
```

```
);


-- Example queries demonstrating relationships

-- Find employees in a specific department (one-to-many)

SELECT FirstName, LastName, JobTitle

FROM Employees

WHERE DeptID = 1;


-- Find projects for a department (one-to-many)

SELECT ProjectName

FROM Projects

WHERE DeptID = 1;


-- Find employees working on a project (many-to-many)

SELECT e.FirstName, e.LastName

FROM Employees e

JOIN EmployeeProjects ep ON e.EmpID = ep.EmpID

WHERE ep.ProjectID = 1;
```

Section 4: Key Takeaways

Identify entities, attributes, and relationships (one-to-many via foreign keys, many-to-many via junction table with composite keys). Use primary keys from "one" side as foreign keys in "many" side. Start with small datasets for SQL practice on companyDB tables: Departments, Employees, Projects, EmployeeProjects. This design supports querying employee-department-project assignments.