Lecture Notes – Normalization

Section 1: Lecture Summary

Normalization is a database design technique involving decomposition of data to eliminate issues in a single large table. Starting from the company database designed via ER diagrams with four tables (Employees, Departments, Projects, EmployeeProjects), the lecture demonstrates problems of combining all data into one table and shows how normalization resolves them through progressive normal forms, leading back to separate tables and the need for joins.

Section 2: Key Concepts and Explanations

Normalization is decomposition to organize data efficiently. Combining all company data (employees, departments, projects) into one table causes **redundancy** (unwanted duplication like repeated department names and locations) and **repeating groups** (multiple values in a single cell, such as multiple project IDs per row). Redundancy leads to update anomalies: changing a department location requires updates in multiple rows, risking **inconsistency** if missed. Repeating groups hinder data retrieval as a cell yields a list instead of a single value.

First normal form (1NF) requires no repeating groups: each cell holds one value, achieved by creating separate rows for each value (e.g., split rows with multiple projects into individual project rows per employee).

Partial dependency occurs when data depends on only part of a composite key (e.g., project details depend partly on department, partly on employee). **Second normal form (2NF)** eliminates partial dependencies by separating such data into dedicated tables (e.g., Projects table for project-department links, EmployeeProjects for employee-project relationships).

Transitive dependency exists when non-key attributes depend on other non-key attributes indirectly (e.g., employee details depend on DeptID, which determines DeptName and Location). **Third normal form (3NF)** removes transitive dependencies by isolating them (e.g., separate Departments table). BCNF is similar but stricter.

Normalization yields the same four-table structure as ER modeling: Employees (EmpID, FirstName, LastName, JobTitle, DeptID, HireDate, Salary), Departments (DeptID, DeptName, Location), Projects (ProjectID, ProjectName, DeptID), EmployeeProjects (EmpID, ProjectID). Decomposition prevents redundancy and anomalies but requires **joins** to reconstruct views across tables for company-wide queries.

Section 3: Example Code and Use Cases

```
USE companyDB;


-- Illustrate unnormalized table issues (conceptual single table with redundancy)

-- Hypothetical view mimicking combined data with repeating groups and redundancy

SELECT

   e.EmpID, e.FirstName, e.LastName, e.JobTitle,

   e.DeptID, d.DeptName, d.Location,  -- Redundancy: DeptName/Location repeat per employee in DeptID

   p.ProjectID, p.ProjectName  -- Repeating groups if multiple projects per row (avoid in real design)

FROM Employees e

JOIN Departments d ON e.DeptID = d.DeptID

JOIN EmployeeProjects ep ON e.EmpID = ep.EmpID

JOIN Projects p ON ep.ProjectID = p.ProjectID

ORDER BY e.EmpID, p.ProjectID;


-- 1NF: Ensure atomic values (already satisfied in schemas; no multi-value cells)

-- Example query assuming normalized split rows for projects

SELECT EmpID, FirstName, LastName, DeptID, ProjectID

FROM Employees e

JOIN EmployeeProjects ep ON e.EmpID = ep.EmpID;  -- One row per employee-project pair
```

```
-- 2NF: Separate Projects (partial dependency removed)
SELECT ProjectID, ProjectName, DeptID
FROM Projects;  -- Projects depend fully on ProjectID + DeptID

-- 3NF: Separate Departments (transitive dependency removed)
SELECT DeptID, DeptName, Location
FROM Departments;  -- DeptName/Location depend only on DeptID

-- Join to reconstruct (reason for normalization: view as single entity)
SELECT e.FirstName, e.LastName, d.DeptName, p.ProjectName
FROM Employees e
JOIN Departments d ON e.DeptID = d.DeptID
JOIN EmployeeProjects ep ON e.EmpID = ep.EmpID
JOIN Projects p ON ep.ProjectID = p.ProjectID
WHERE d.DeptName = 'Human Resource';
```

Section 4: Key Takeaways

Normalization decomposes a single redundant table into multiple related tables to avoid redundancy, inconsistency, and repeating groups. 1NF eliminates repeating groups, 2NF removes partial dependencies, 3NF/BCNF eliminates transitive dependencies. Results match ER modeling (four companyDB tables). Separate tables require joins to query the full company database as a unified view. Practice with examples is essential for mastery.