Lecture Notes – Complex & Materialized View

Section 1: Lecture Summary

The lecture covers **complex views** and **materialized views** using the companyDB. It demonstrates creating a complex view for department salary summary by joining Departments and Employees tables, grouping by department, and using aggregate functions for total salary and employee count. Materialized views store query results physically for faster access when data changes infrequently.

Section 2: Key Concepts and Explanations

Simple view: Created from a single table.

Complex view: Joins multiple tables (e.g., Departments and Employees), uses **GROUP BY**, and aggregate functions like **SUM()** and **COUNT()** to produce summary data not directly matching any base table. The view executes the query each time it is queried, retrieving fresh data from underlying tables.

Materialized view: Stores the query result as a physical table for performance. Use when summary data updates rarely (e.g., yearly) but is queried frequently, avoiding repeated execution of complex joins and aggregations. Note: Materialized views work in PostgreSQL (e.g., PGAdmin), not MySQL Workbench.

Views simplify queries: Users reference the view name instead of rewriting complex logic, and access is limited to view columns.

Section 3: Example Code and Use Cases

Creating the complex view (department salary summary from companyDB):

```
CREATE VIEW dept_salary_summary AS
SELECT
    d.DeptID,
    d.DeptName,
    SUM(e.Salary) AS total_dept_salary,
    COUNT(e.EmpID) AS emp_count
FROM Departments d
NATURAL JOIN Employees e
GROUP BY d.DeptID, d.DeptName;
```

Using the complex view:

```
SELECT * FROM dept_salary_summary;
```

This executes the underlying join and aggregation, displaying DeptID, DeptName, total_dept_salary (e.g., 220000 for Human Resources), and emp_count (e.g., 4).

Converting to materialized view (conceptual; syntax for PostgreSQL):

```
CREATE MATERIALIZED VIEW mv_dept_salary_summary AS
SELECT
    d.DeptID,
    d.DeptName,
    SUM(e.Salary) AS total_dept_salary,
    COUNT(e.EmpID) AS emp_count
FROM Departments d
NATURAL JOIN Employees e
GROUP BY d.DeptID, d.DeptName;
```

Query it as: `SELECT * FROM mv_dept_salary_summary;` (faster, uses stored results).

Section 4: Key Takeaways

Complex views handle multi-table joins and aggregations for summaries. Materialized views store results for speed when data is static. Use views to simplify repeated complex queries and restrict access to summary data.