Lecture Notes – What Are Joins

Section 1: Lecture Summary

The lecture introduces **joins** as a method for fetching data from multiple tables in SQL queries. It explains that listing multiple tables in the FROM clause without a JOIN keyword results in a **cross join**, which cross-multiplies all rows from each table, producing a large result set. Adding a WHERE clause to match common columns, such as **DeptID**, filters the results to show only matching rows, similar to how joins function but with different syntax.

Section 2: Key Concepts and Explanations

Joins retrieve data from multiple tables by combining rows based on conditions. Types of joins include **cross join**, **inner join**, **natural join**, **left join** (left outer join), **right join** (right outer join), **full outer join**, and **self join**. Listing multiple tables in FROM (e.g., Employees, Departments) without JOIN creates a **cross join**, pairing every row from the first table with every row from the second, resulting in M × N rows (e.g., 20 Employees × 4 Departments = 80 rows). **DeptID** serves as primary key in Departments and foreign key in Employees, enabling matching. Adding WHERE Employees.DeptID = Departments.DeptID filters to matching rows only, yielding max(M, N) rows (e.g., 20 rows). Joins use equivalent logic but replace comma with JOIN keyword and WHERE with ON clause for clarity.

Section 3: Example Code and Use Cases

Using companyDB schemas:

Cross join behavior (multiple tables in FROM):

```
SELECT * FROM Employees, Departments;
```

Result: 80 rows, all columns from both tables (**EmpID**, **FirstName**, **LastName**, **JobTitle**, **DeptID**, **HireDate**, **Salary**, **DeptID**, **DeptName**, **Location**), with **DeptID** duplicated; every Employees row paired with every Departments row.

Filtered cross join (matching **DeptID**):

```
SELECT * FROM Employees, Departments
WHERE Employees.DeptID = Departments.DeptID;
```

Result: 20 rows, only matching pairs (e.g., Employees row with DeptID=1 paired with Departments row DeptID=1).

Equivalent join syntax (introduced for next lecture):

```
SELECT * FROM Employees
JOIN Departments ON Employees.DeptID = Departments.DeptID;
```

Section 4: Key Takeaways

Multiple tables in FROM clause default to **cross join** (cross-multiplication). Use WHERE on matching columns like **DeptID** to filter to relevant rows. Joins follow the same process but use explicit JOIN ... ON syntax for readability. Understanding this foundation applies to all join types.