Lecture Notes – Left,Right & Full Join Queries

Section 1: Lecture Summary

This lecture covers **LEFT JOIN**, **RIGHT JOIN**, and **FULL JOIN** operations in SQL, demonstrating how each join type handles matching and non-matching records differently. The instructor explains that these outer joins preserve unmatched rows from one or both tables, unlike inner joins which only return matching records. Practical examples are provided using real database schemas to illustrate when each join type is appropriate.

Section 2: Key Concepts and Explanations

LEFT JOIN (Left Outer Join)

A LEFT JOIN returns all records from the left table and matching records from the right table. Unmatched rows from the left table appear with NULL values for right table columns. This is useful when you need all records from the primary table regardless of whether matches exist in the secondary table.

RIGHT JOIN (Right Outer Join)

A RIGHT JOIN returns all records from the right table and matching records from the left table. Unmatched rows from the right table appear with NULL values for left table columns. The order of tables in the FROM and JOIN clauses determines which table is considered "left" and which is "right."

FULL JOIN (Full Outer Join)

A FULL JOIN returns all records from both tables. Unmatched rows from either table appear with NULL values for columns from the non-matching table. This join type is not supported in MySQL but is available in other databases like PostgreSQL.

Key Difference: Table Order Matters

The position of tables in your query determines join behavior. When you write `FROM table1 LEFT JOIN table2`, table1 is the left table. If you reverse the order, the join behaves differently even if the keyword remains the same.

Section 3: Example Code and Use Cases

Use Case 1: Show All Customers and Their Orders

This scenario requires retrieving every customer in the database along with their orders, including customers who have never placed an order.

```
SELECT
    c.CustomerID,
    c.FirstName,
    c.LastName,
    o.OrderID,
    o.OrderDate
FROM Customers c
LEFT JOIN Orders o
    ON c.CustomerID = o.CustomerID;
```

This query returns all 15 customers from the Customers table. For customers without orders (CustomerID 11, 12, 13, 14, 15), the OrderID and OrderDate columns display NULL values.

Using RIGHT JOIN for the Same Data

```
SELECT
    c.CustomerID,
```

```
    c.FirstName,
    c.LastName,
    o.OrderID,
    o.OrderDate
FROM Customers c
RIGHT JOIN Orders o
    ON c.CustomerID = o.OrderID;
```

This returns only customers who have placed orders, since every order in the Orders table is associated with an existing customer. No NULL values appear because the join fills all rows with matching customer data.

Use Case 2: List All Reviews Written by Customers

```
SELECT
    c.CustomerID,
    c.FirstName,
    c.LastName,
    r.Rating,
    r.ReviewDate
FROM Reviews r
LEFT JOIN Customers c
    ON r.CustomerID = c.CustomerID;
```

This query returns all reviews in the Reviews table with corresponding customer information. Since every review is written by some customer, all rows contain complete data with no NULL values.

Reverse with RIGHT JOIN

```
SELECT
    c.CustomerID,
    c.FirstName,
    c.LastName,
    r.Rating,
    r.ReviewDate
```

```
FROM Reviews r
RIGHT JOIN Customers c
    ON r.CustomerID = c.CustomerID;
```

This returns all customers with their reviews. Customers who have not written any review show NULL values for Rating and ReviewDate columns.

FULL JOIN Alternative Using UNION

Since MySQL does not support FULL JOIN, you can achieve the same result using LEFT JOIN and RIGHT JOIN combined with UNION:

```
SELECT
    c.CustomerID,
    c.FirstName,
    c.LastName,
    o.OrderID,
    o.OrderDate
FROM Customers c
LEFT JOIN Orders o
    ON c.CustomerID = o.CustomerID

UNION

SELECT
    c.CustomerID,
    c.FirstName,
    c.LastName,
    o.OrderID,
    o.OrderDate
FROM Customers c
RIGHT JOIN Orders o
    ON c.CustomerID = o.CustomerID;
```

This returns all customers and all orders, with NULL values for unmatched records from both tables.

Section 4: Key Takeaways

The choice between LEFT JOIN, RIGHT JOIN, and FULL JOIN depends on which table's records you want to preserve. LEFT JOIN preserves all left table records, RIGHT JOIN preserves all right table records, and FULL JOIN preserves all records from both tables. MySQL does not natively support FULL JOIN, but you can simulate it using UNION with LEFT and RIGHT JOINs. When writing queries, carefully consider table order because the position of tables in the FROM clause determines which table is "left" and which is "right." Outer joins are essential for retrieving complete datasets that include unmatched records, which regular INNER JOINs cannot provide.