Database Fundamentals

Cybersecurity Department

Course Code: ***CBS 213***

Practical Lecture 7: Advanced SQL – JOINs

Halal Abdulrahman Ahmed

Outlines

- Why JOINs are needed?
- Types of JOINs.

Learning Outcomes

By the end of this practical lecture, students should be able to:

- Explain why JOINs are needed in normalized databases.
- Write and run **INNER, LEFT, RIGHT** joins and interpret their results.
- Implement **FULL JOIN behavior in MySQL** using UNION.
- Use **NATURAL JOIN** and describe when it is safe/unsafe.

SQL Joins (Inner, Left, Right and Full Join)

In relational databases:

a. Data is **split across multiple tables** for normalization

b. Each table holds **different information**

c. JOINs combine related data using **keys**

- SQL joins are fundamental tools for combining data from multiple tables in relational databases.

- Normalization **creates** relationships. JOIN **uses** those relationships to retrieve data.

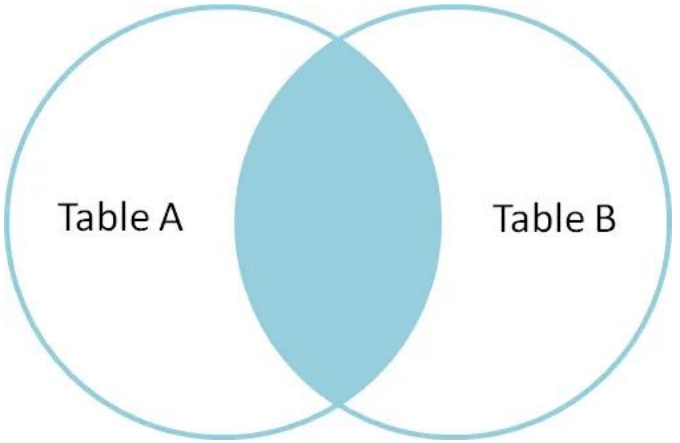
Types of SQL Joins

1. SQL INNER JOIN

The INNER JOIN keyword selects all rows from both the tables as long as the condition is satisfied. This keyword will create the result set by combining all rows from both the tables where the condition satisfies i.e value of the common field will be the same.

Syntax:

```
SELECT table1.column1, table1.column2, table2.column1, ...
FROM table1
INNER JOIN table2
ON table1.matching_column = table2.matching_column;
```



- We can also write JOIN instead of INNER JOIN. JOIN is same as INNER JOIN.

Example of INNER JOIN

```
CREATE DATABASE IF NOT EXISTS CollegeDB;
USE CollegeDB;
```

Create database

```
CREATE TABLE Student (
    ROLL_NO INT AUTO_INCREMENT,
    NAME VARCHAR(50),
    ADDRESS VARCHAR(50),
    PHONE VARCHAR(15),
    AGE INT,
    PRIMARY KEY (ROLL_NO)
);
```

Create table

Example of INNER JOIN

```
INSERT INTO Student (NAME, ADDRESS, PHONE, AGE) VALUES
('HARSH', 'DELHI', 'XXXXXXXXXX', 18),
('PRATIK', 'BIHAR', 'XXXXXXXXXX', 19),
('RIYANKA', 'SILIGURI', 'XXXXXXXXXX', 20),
('DEEP', 'RAMNAGAR', 'XXXXXXXXXX', 18),
('ANITA', 'MUMBAI', 'XXXXXXXXXX', 21);
```

Insert Data into Student

```sql
CREATE TABLE StudentCourse (
    COURSE_ID INT,
    ROLL_NO INT,
    FOREIGN KEY (ROLL_NO) REFERENCES Student(ROLL_NO)
);
```

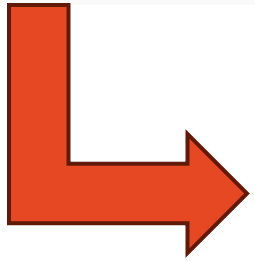
Create StudentCourse Table

```sql
INSERT INTO StudentCourse (COURSE_ID, ROLL_NO) VALUES
(1, 1),
(2, 2),
(3, 3);
```

Insert Data into StudentCourse

Example of INNER JOIN

```
SELECT StudentCourse.COURSE_ID, Student.NAME, Student.AGE
FROM Student
INNER JOIN StudentCourse
ON Student.ROLL_NO = StudentCourse.ROLL_NO;
```



INNER JOIN Query

Output

COURSE_ID	NAME	AGE
1	HARSH	18
2	PRATIK	19
3	RIYANKA	20

Output

COURSE_ID	NAME	AGE
1	HARSH	18
2	PRATIK	19
3	RIYANKA	20

Students **with courses** → shown

Students **without courses → excluded (not shown at all)**

INNER JOIN = **common records only**

- **INNER JOIN Result – What does it mean?**
- **“Shows only matching rows”**
- A row is shown **only if it exists in BOTH tables**
- If a student **does not have a course**, they are **removed from the result**

2. SQL LEFT JOIN

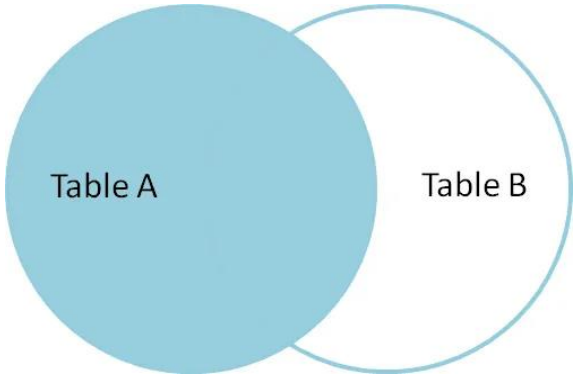
A LEFT JOIN returns all rows from the left table, along with matching rows from the right table. If there is no match, NULL values are returned for columns from the right table. LEFT JOIN is also known as LEFT OUTER JOIN.

- LEFT JOIN Syntax

```
SELECT table1.column1, table1.column2, table2.column1, ...
FROM table1
LEFT JOIN table2
ON table1.matching_column = table2.matching_column;
```

- We can also use LEFT OUTER JOIN instead of LEFT JOIN, both are the same.

```
LEFT JOIN
LEFT OUTER JOIN
```



Example of LEFT JOIN

```
SELECT Student.NAME, StudentCourse.COURSE_ID
FROM Student
LEFT JOIN StudentCourse
ON Student.ROLL_NO = StudentCourse.ROLL_NO;
```

Output

NAME	COURSE_ID
HARSH	1
PRATIK	2
RIYANKA	3
DEEP	NULL
ANITA	NULL

LEFT JOIN Result "Shows all students"

- Every student from the **Student table** is shown
- Even if a student **has no course**

"Students without courses included with NULL"

- The student **exists b**ut there is **no matching record** in StudentCourse
- So course-related columns show **NULL**
- NULL means **no matching data found**

3. SQL RIGHT JOIN

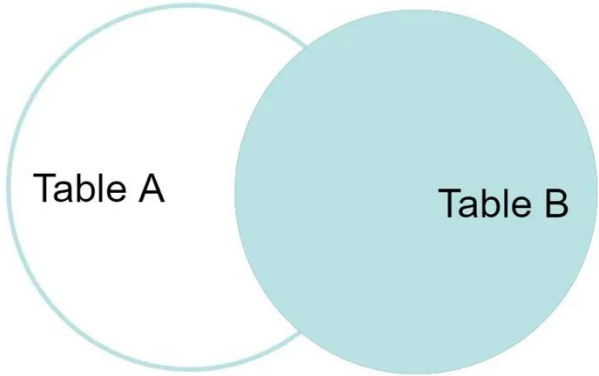
RIGHT JOIN returns all the rows of the table on the right side of the join and matching rows for the table on the left side of the join. It is very similar to LEFT JOIN for the rows for which there is no matching row on the left side, the result-set will contain null. RIGHT JOIN is also known as RIGHT OUTER JOIN.

RIGHT JOIN Syntax

```
SELECT table1.column1,table1.column2,table2.column1,....
FROM table1
RIGHT JOIN table2
ON table1.matching_column = table2.matching_column;
```

- RIGHT JOIN is also called RIGHT OUTER JOIN. Both are the same.

```
LEFT JOIN
LEFT OUTER JOIN
```



Example of RIGHT JOIN

```
SELECT Student.NAME, StudentCourse.COURSE_ID
FROM Student
RIGHT JOIN StudentCourse
ON StudentCourse.ROLL_NO = Student.ROLL_NO;
```

Output

NAME	COURSE_ID
HARSH	1
PRATIK	2
RIYANKA	3
NULL	6

Why does NULL appear?

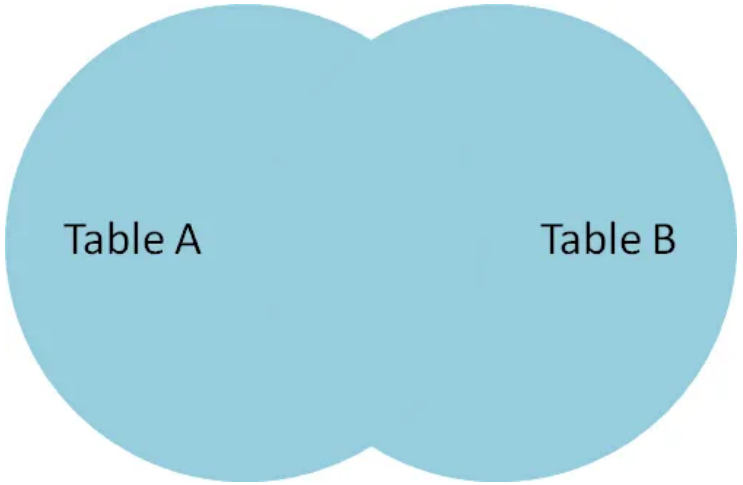
- COURSE_ID = 6 exists in **StudentCourse**
- Its ROLL_NO does **not exist** in **Student**
- RIGHT JOIN keeps **all rows from the right table (StudentCourse)**
- Missing match → NULL in left table columns (Student.NAME)

4. SQL FULL JOIN

- **FULL JOIN** creates the result-set by combining results of both LEFT JOIN and RIGHT JOIN. The result-set will contain all the rows from both tables. For the rows for which there is no matching, the result-set will contain NULL values.

Syntax

```
SELECT table1.column1,table1.column2,table2.column1,....
FROM table1
FULL JOIN table2
ON table1.matching_column = table2.matching_column;
```



Example of FULL JOIN

```
SELECT Student.NAME, StudentCourse.COURSE_ID
FROM Student
LEFT JOIN StudentCourse
ON Student.ROLL_NO = StudentCourse.ROLL_NO
UNION
SELECT Student.NAME, StudentCourse.COURSE_ID
FROM Student
RIGHT JOIN StudentCourse
ON Student.ROLL_NO = StudentCourse.ROLL_NO;
```

Output

NAME	COURSE_ID
HARSH	1
PRATIK	2
RIYANKA	3
DEEP	NULL
ANITA	NULL
NULL	6

- Students **with courses** → shown normally
- Students **without courses** → shown with NULL
- Courses **without students** → shown with NULL

5. SQL Natural Join

- A Natural Join is a type of INNER JOIN that automatically joins two tables based on columns with the same name and data type. It returns only the rows where the values in the common columns match.
- It returns rows where the values in these common columns are the same in both tables.
- Common columns appear only once in the result, even if they exist in both tables.
- Unlike a CROSS JOIN, which creates all possible combinations of rows, a Natural Join only includes rows with matching values

Example of Natural JOIN

```
SELECT *
FROM Student
NATURAL JOIN StudentCourse;
```

Output

ROLL_NO	NAME	ADDRESS	PHONE	AGE	COURSE_ID
1	HARSH	DELHI	XXXXXXXXXX	18	1
2	PRATIK	BIHAR	XXXXXXXXXX	19	2
3	RIYANKA	SILIGURI	XXXXXXXXXX	20	3

- ROLL_NO **4 and 5** exist only in Student
- ROLL_NO **7** exists only in StudentCourse
- NATURAL JOIN returns **only matching rows**

Full Example to Implement Joins

```sql
/* ---------- Block 1: Reset + Create Database ---------- */
DROP DATABASE IF EXISTS CollegeDB;
CREATE DATABASE CollegeDB;
USE CollegeDB;

/* ---------- Block 2: Create Student Table ---------- */
CREATE TABLE Student (
    ROLL_NO INT AUTO_INCREMENT,
    NAME VARCHAR(50) NOT NULL,
    ADDRESS VARCHAR(50),
    PHONE VARCHAR(15),
    AGE INT,
    PRIMARY KEY (ROLL_NO)
);
```

Full Example to Implement Joins

```sql
/* Block 3: Insert Data into Student */
INSERT INTO Student (NAME, ADDRESS, PHONE, AGE) VALUES
('HARSH', 'DELHI', 'XXXXXXXXXX', 18),
('PRATIK', 'BIHAR', 'XXXXXXXXXX', 19),
('RIYANKA', 'SILIGURI', 'XXXXXXXXXX', 20),
('DEEP', 'RAMNAGAR', 'XXXXXXXXXX', 18),
('ANITA', 'MUMBAI', 'XXXXXXXXXX', 21);

/* Block 4: Create StudentCourse Table */
CREATE TABLE StudentCourse (
    COURSE_ID INT NOT NULL,
    ROLL_NO INT NOT NULL
);
```

Full Example to Implement Joins

```sql
/* Block 5: Insert Data into StudentCourse */
INSERT INTO StudentCourse (COURSE_ID, ROLL_NO) VALUES
(1, 1),
(2, 2),
(3, 3),
(6, 7);

/* Block 6: INNER JOIN */
SELECT
    StudentCourse.COURSE_ID,
    Student.NAME,
    Student.AGE
FROM Student
INNER JOIN StudentCourse
ON Student.ROLL_NO = StudentCourse.ROLL_NO;
```

Full Example to Implement Joins

```sql
/* Block 7: LEFT JOIN */
SELECT
    Student.NAME,
    StudentCourse.COURSE_ID
FROM Student
LEFT JOIN StudentCourse
ON Student.ROLL_NO = StudentCourse.ROLL_NO;

/* Block 8: RIGHT JOIN */
SELECT
    Student.NAME,
    StudentCourse.COURSE_ID
FROM Student
RIGHT JOIN StudentCourse
ON Student.ROLL_NO = StudentCourse.ROLL_NO;
```

Full Example to Implement Joins

```sql
/* Block 9: FULL JOIN (MySQL workaround using UNION) */
SELECT
    Student.NAME,
    StudentCourse.COURSE_ID
FROM Student
LEFT JOIN StudentCourse
ON Student.ROLL_NO = StudentCourse.ROLL_NO
UNION
SELECT
    Student.NAME,
    StudentCourse.COURSE_ID
FROM Student
RIGHT JOIN StudentCourse
ON Student.ROLL_NO = StudentCourse.ROLL_NO;
```

Full Example to Implement Joins

```sql
/* Block 10: NATURAL JOIN */
SELECT *
FROM Student
NATURAL JOIN StudentCourse;
```

Homework

- Difference Between INNER JOIN and NATURAL JOIN

References

- Budy, S., Reese, G., & Tahaghoghi, S. M. M. (Year unknown). *Learning MySQL*. O'Reilly Media.
- Beaulieu, A. (Year unknown). *MySQL Cookbook* (4th ed.). O'Reilly Media.
- GeeksforGeeks. (2025, November 7). *SQL Joins (Inner, Left, Right and Full Join)*. GeeksforGeeks. https://www.geeksforgeeks.org/sql/sql-join-set-1-inner-left-right-and-full-joins/


Any
Question