Week-7

Introduction to Database Systems

Shah Nawaz

SQL Joins

Joins in SQL

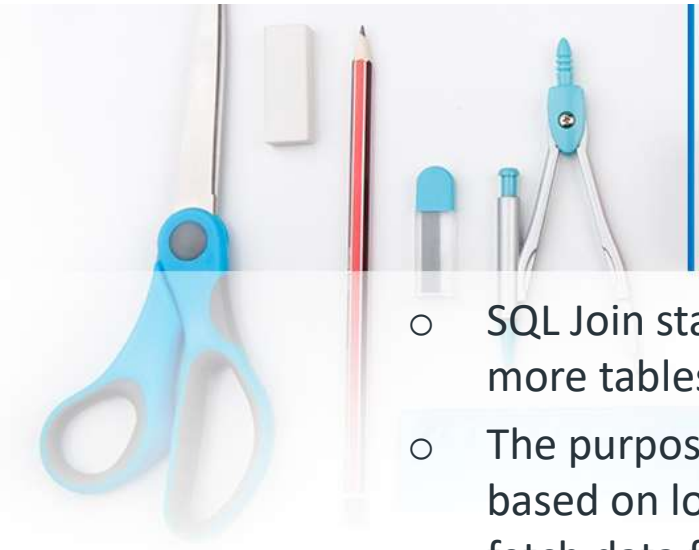
- **Need for Joins**

Student

RollNumber	Name	Did
S1	Ahmad	D1
S2	Aliya	D1
S3	Bushra	D2
S4	Bilawal	D2
S5	Sameena	NULL
S6	Seher	NULL

Department

DNumber	Dname
D1	FOIT
D2	FOEE
D3	FOSS



Joins in SQL

- SQL Join statement is used to combine data or rows from two or more tables based on a common field between them.
- The purpose of JOINs in SQL is to access data from multiple tables based on logical relationships between them. JOINS are used to fetch data from database tables and represent the result dataset as a separate table.

Different types of Joins are as follows:

1. **INNER JOIN**
2. **NATURAL JOIN**
3. **CROSS JOIN**
4. **OUTER JOIN**
    a. **LEFT JOIN**
    b. **RIGHT JOIN**
    c. **FULL JOIN**

Inner Join

An SQL INNER JOIN is used to combine rows from two or more tables based on a related column between them. The result set includes only the rows that have matching values in the specified columns. The syntax for an INNER JOIN is as follows:

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

Here's a breakdown of the syntax:

- **SELECT column1, column2, ...**: Specifies the columns you want to retrieve from the tables.
- **FROM table1**: Specifies the first table in the join operation.
- **INNER JOIN table2**: Specifies the second table in the join operation.
- **ON table1.column_name = table2.column_name**: Specifies the condition for the join, indicating which columns from each table should be used for the matching.

Inner Join

Student

RollNumber	Name	Did
S1	Ahmad	D1
S2	Aliya	D1
S3	Bushra	D2
S4	Bilawal	D2
S5	Sameena	NULL
S6	Seher	NULL

Department

DNumber	Dname
D1	FOIT
D2	FOEE
D3	FOSS

```
SELECT Student.RollNumber, Student.Name , Department.Dname
FROM Student
INNER JOIN Department
ON  Student.Did= Department.DNumber;
```

Output: Student Inner Join Department

RollNumber	Name	Dname
S1	Ahmad	FOIT
S2	Aliya	FOIT
S3	Bushra	FOEE
S4	Bilawal	FOEE

Natural Join

A NATURAL JOIN combines rows from two or more tables based on columns with the same name and data types.

It automatically matches and combines columns with identical names between the tables.

Don't explicitly specify the columns for comparison in a NATURAL JOIN—the database engine automatically determines the matching columns.

```
SELECT *
FROM table1
NATURAL JOIN table2;
```

In this example, the database engine identifies the columns with the same name in both tables and performs the join based on those columns. It's important to note that if there are multiple columns with the same name in both tables, the join is performed on all of them, creating an implicit "AND" condition for each matching pair of columns.

Natural Join

Student

RollNumber	Name	Did
S1	Ahmad	D1
S2	Aliya	D1
S3	Bushra	D2
S4	Bilawal	D2
S5	Sameena	NULL
S6	Seher	NULL

Department

Did	Dname
D1	FOIT
D2	FOEE
D3	FOSS

```
SELECT *
FROM STUDENT
NATURAL JOIN Department;
```

Output: Student Inner Join Department

Rollnumber	Name	Did	Dname
S1	Ahmad	D1	FOIT
S2	Aliya	D1	FOIT
S3	Bushra	D2	FOEE
S4	Bilawal	D2	FOEE

Cross Join

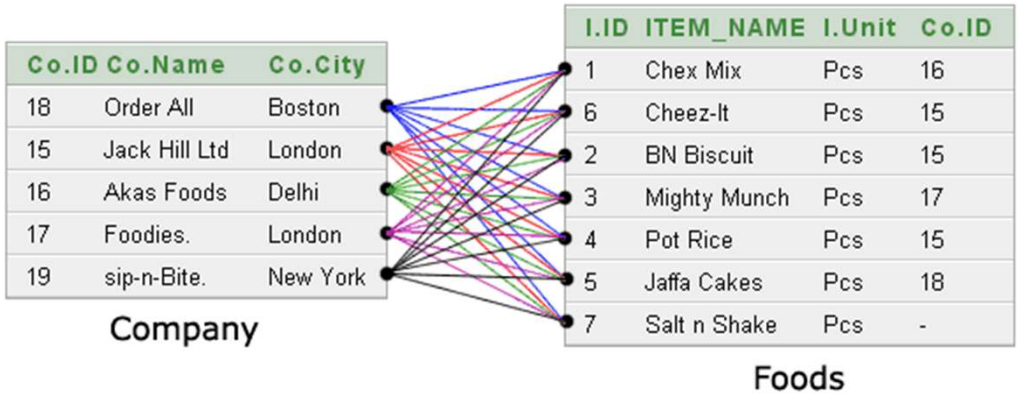
The CARTESIAN JOIN or CROSS JOIN returns the Cartesian product of the sets of records from two or more joined tables.

Thus, it equates to an inner join where the join-condition always evaluates to either True or where the join-condition is absent from the statement.

```
SELECT *
FROM table1
CROSS JOIN table2;
```

Cross Join

```
SELECT *
FROM Company
CROSS JOIN Foods;
```



Co.ID	Co.Name	Co.City
18	Order All	Boston
15	Jack Hill Ltd	London
16	Akas Foods	Delhi
17	Foodies.	London
19	sip-n-Bite.	New York

Company

I.ID	ITEM_NAME	I.Unit	Co.ID
1	Chex Mix	Pcs	16
6	Cheez-It	Pcs	15
2	BN Biscuit	Pcs	15
3	Mighty Munch	Pcs	17
4	Pot Rice	Pcs	15
5	Jaffa Cakes	Pcs	18
7	Salt n Shake	Pcs	-

Foods

Database engine will pick first tuple of Company table and make combinations with all 7 records of Foods table. Then Database engine will second record of Company and make combinations with all 7 records of Foods table and so on…

Total records in output table will be :

Number of tuples of table 1 X number of tuples of table 2

In this example, resultant table will comprise of 5 x 7 = 35 tuples

Cross Join

```
SELECT *
FROM Company
CROSS JOIN Foods;
```

Co.ID	Co.Name	Co.City
18	Order All	Boston
15	Jack Hill Ltd	London
16	Akas Foods	Delhi
17	Foodies.	London
19	sip-n-Bite.	New York

Company

I.ID	ITEM_NAME	I.Unit	Co.ID
1	Chex Mix	Pcs	16
6	Cheez-It	Pcs	15
2	BN Biscuit	Pcs	15
3	Mighty Munch	Pcs	17
4	Pot Rice	Pcs	15
5	Jaffa Cakes	Pcs	18
7	Salt n Shake	Pcs	-

Foods

Co.ID_x	Co.Name	Co.City	I.ID	ITEM_NAME	I.Unit	Co.ID_y
18	Order All	Boston	1	Chex Mix	Pcs	16
18	Order All	Boston	6	Cheez-It	Pcs	15
18	Order All	Boston	2	BN Biscuit	Pcs	15
18	Order All	Boston	3	Mighty Munch	Pcs	17
18	Order All	Boston	4	Pot Rice	Pcs	15
18	Order All	Boston	5	Jaffa Cakes	Pcs	18
18	Order All	Boston	7	Salt n Shake	Pcs	
15	Jack Hill Ltd	London	1	Chex Mix	Pcs	16
15	Jack Hill Ltd	London	6	Cheez-It	Pcs	15
15	Jack Hill Ltd	London	2	BN Biscuit	Pcs	15
15	Jack Hill Ltd	London	3	Mighty Munch	Pcs	17
15	Jack Hill Ltd	London	4	Pot Rice	Pcs	15
15	Jack Hill Ltd	London	5	Jaffa Cakes	Pcs	18
15	Jack Hill Ltd	London	7	Salt n Shake	Pcs	
16	Akas Foods	Delhi	1	Chex Mix	Pcs	16
16	Akas Foods	Delhi	6	Cheez-It	Pcs	15
16	Akas Foods	Delhi	2	BN Biscuit	Pcs	15
16	Akas Foods	Delhi	3	Mighty Munch	Pcs	17
16	Akas Foods	Delhi	4	Pot Rice	Pcs	15
16	Akas Foods	Delhi	5	Jaffa Cakes	Pcs	18
16	Akas Foods	Delhi	7	Salt n Shake	Pcs	
17	Foodies.	London	1	Chex Mix	Pcs	16
17	Foodies.	London	6	Cheez-It	Pcs	15
17	Foodies.	London	2	BN Biscuit	Pcs	15
17	Foodies.	London	3	Mighty Munch	Pcs	17
17	Foodies.	London	4	Pot Rice	Pcs	15
17	Foodies.	London	5	Jaffa Cakes	Pcs	18
17	Foodies.	London	7	Salt n Shake	Pcs	
19	sip-n-Bite.	New York	1	Chex Mix	Pcs	16
19	sip-n-Bite.	New York	6	Cheez-It	Pcs	15
19	sip-n-Bite.	New York	2	BN Biscuit	Pcs	15
19	sip-n-Bite.	New York	3	Mighty Munch	Pcs	17
19	sip-n-Bite.	New York	4	Pot Rice	Pcs	15
19	sip-n-Bite.	New York	5	Jaffa Cakes	Pcs	18
19	sip-n-Bite.	New York	7	Salt n Shake	Pcs	

Outer Join

- An OUTER JOIN in SQL is used to retrieve rows from one or more tables, even if there is no match in the joined table(s).
- It includes unmatched rows from one or both tables in the result set, filling in the gaps with NULL values for columns where no match is found.

There are three types of OUTER JOINs:

- **LEFT OUTER JOIN,**
- **RIGHT OUTER JOIN, and**
- **FULL OUTER JOIN.**

Left Outer Join

Returns all rows from the left table and the matched rows from the right table. If there is no match in the right table, NULL values are returned for columns from the right table.

```
SELECT *
FROM table1
LEFT OUTER JOIN table2
ON table1.column_name = table2.column_name;
```

```
SELECT *
FROM left_table
LEFT OUTER JOIN right_table
ON left_table.column_name = right_table.column_name;
```

Important point

Left Outer Join

Student

RollNumber	Name	Did
S1	Ahmad	D1
S2	Aliya	D1
S3	Bushra	D2
S4	Bilawal	D2
S5	Sameena	NULL
S6	Seher	NULL

Department

DNumber	Dname
D1	FOIT
D2	FOEE
D3	FOSS

```
SELECT Student.RollNumber, Student.Name , Department.Dname
FROM Student
LEFT OUTER JOIN Department ON  Student.Did= Department.DNumber;
```

Output:

RollNumber	Name	Dname
S1	Ahmad	FOIT
S2	Aliya	FOIT
S3	Bushra	FOEE
S4	Bilawal	FOEE
S5	Sameena	NULL
S6	Seher	NULL

Right Outer Join

Returns all rows from the right table and the matched rows from the left table. If there is no match in the left table, NULL values are returned for columns from the left table.

```
SELECT *
FROM table1
RIGHT OUTER JOIN table2
ON table1.column_name = table2.column_name;
```

Right Outer Join

Student

RollNumber	Name	Did
S1	Ahmad	D1
S2	Aliya	D1
S3	Bushra	D2
S4	Bilawal	D2
S5	Sameena	NULL
S6	Seher	NULL

Department

DNumber	Dname
D1	FOIT
D2	FOEE
D3	FOSS

```
SELECT Student.RollNumber, Student.Name , Department.Dname
FROM Student
RIGHT OUTER JOIN Department ON  Student.Did= Department.DNumber;
```

Output:

RollNumber	Name	Dname
S1	Ahmad	FOIT
S2	Aliya	FOIT
S3	Bushra	FOEE
S4	Bilawal	FOEE
NULL	NULL	FOSS

Full Outer Join

Returns all rows when there is a match in either the left or right table. If there is no match in one of the tables, NULL values are returned for columns from the table without a match.

```
SELECT *
FROM table1
FULL OUTER JOIN table2
ON table1.column_name = table2.column_name;
```

Full Outer Join

Student

RollNumber	Name	Did
S1	Ahmad	D1
S2	Aliya	D1
S3	Bushra	D2
S4	Bilawal	D2
S5	Sameena	NULL
S6	Seher	NULL

Department

DNumber	Dname
D1	FOIT
D2	FOEE
D3	FOSS

```
SELECT Student.RollNumber, Student.Name , Department.Dname
FROM Student
FULL OUTER JOIN Department ON  Student.Did= Department.DNumber;
```

Output:

RollNumber	Name	Dname
S1	Ahmad	FOIT
S2	Aliya	FOIT
S3	Bushra	FOEE
S4	Bilawal	FOEE
S5	Sameena	NULL
S6	Seher	NULL
NULL	NULL	FOSS

Alias

In SQL, an alias is a temporary name that is used to rename a table or a column for the duration of a query.
Aliases are helpful for making the SQL code more readable and concise, especially when dealing with complex queries or when tables and columns have long or ambiguous names.

There are two main types of aliases in SQL:

1.Table Alias:
2.Column Alias:

Alias

1 - Table Aliases:

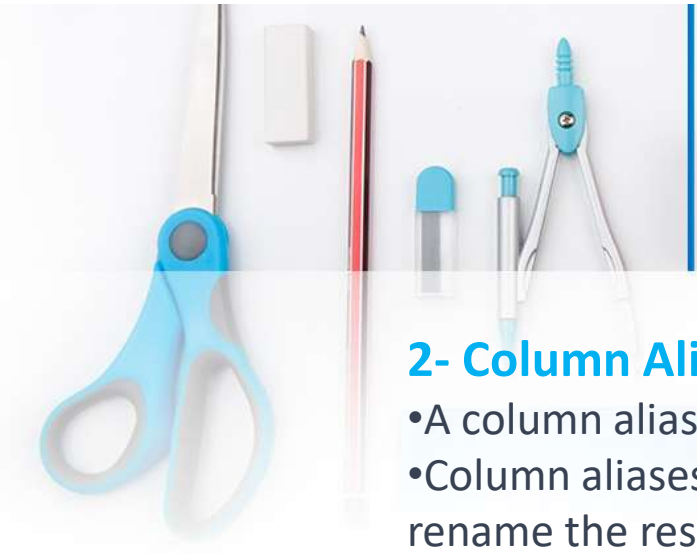
- A table alias is used to provide a temporary, alternative name for a table in a query.
- Table aliases are particularly useful when you are dealing with joins.

Syntax:

```
SELECT column1, column2
FROM table_name AS alias_name;
```

Example:

```
SELECT e.employee_id, e.employee_name, d.department_name
FROM employees AS e
INNER JOIN departments AS d ON e.department_id = d.department_id;
```



Alias

2- Column Aliases:

•A column alias is used to provide a temporary name for a column in the result set.
•Column aliases are often used to make the output of a query more readable or to rename the result of a computation.

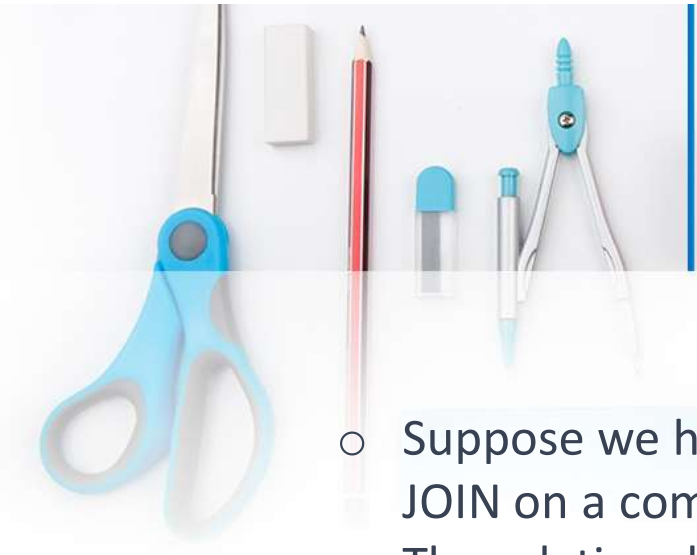
Syntax:

```
SELECT column_name AS alias_name
FROM table_name;
```

Example:

```
SELECT employee_name AS "Emp Name", salary * 12 AS "Annual Salary"
FROM employees;
```

Relational Algebra for Joins



Relational Algebra for Inner Join

⋈ = Bowtie (Bow-tie)

- Suppose we have two tables, ***A*** and ***B***, and we want to perform an INNER JOIN on a common attribute ***C*** where ***C*** is a column in both tables.
- The relational algebra expression for INNER JOIN is:

$$\sigma A.C=B.C(A \bowtie B)$$

Pronunciation: "Selection where A.C equals B.C of the inner join of A and B.“

Explanation:

- $A \bowtie B$ represents the natural join of tables A and B.
- $\sigma A.C=B.C$ represents the selection operation where we filter the rows where the values in column C are equal in both tables A and B.

Relational Algebra for Natural Join

- For NATURAL JOIN, no specific join condition is specified. It automatically joins the tables based on columns with the same name.
- The relational algebra expression for NATURAL JOIN is:

$$\sigma(A \bowtie B)$$

Pronunciation: A bowtie B or Natural join of A and B

Explanation:

- $A \bowtie B$ represents the natural join of tables A and B. In this case, the join is performed based on columns with the same name in both tables.

Relational Algebra for Cross Join

The relational algebra expression for a CROSS JOIN (also known as a Cartesian product) between two tables, A and B, is denoted as follows:

$$\sigma(A \times B)$$

Pronunciation: "A cross B"

Explanation:

- *A* and *B* represent the tables involved in the cross join.
- The × (multiplication) symbol represents the Cartesian product or cross join operation.

Relational Algebra for Outer Join

- Relational algebra does not have explicit operators for LEFT OUTER JOIN, RIGHT OUTER JOIN, or FULL OUTER JOIN as these are specific to SQL.
- However, you can represent these operations in relational algebra using the basic operations of :
  - **selection (σ),**
  - **projection (π), and the**
  - **natural Join ($\bowtie$).**
  - **Left Outer Join ($⟕$)**
  - **Right Outer Join ($⟖$)**
  - **Full Outer Join ($⟗$)**

Relational Algebra for Left Outer Join

```
SELECT Student.RollNumber, Student.Name , Department.Dname
FROM Student
LEFT OUTER JOIN Department ON  Student.Did= Department.DNumber;
```

RollNumber	Name	Dname
S1	Ahmad	FOIT
S2	Aliya	FOIT
S3	Bushra	FOEE
S4	Bilawal	FOEE
S5	Sameena	NULL
S6	Seher	NULL

Assuming you have two tables A and B, and you want to perform a LEFT OUTER JOIN on a condition C:

$$\pi_{\{RollNumber, Name, Dname\}} (Student \bowtie_{\{Student.Did = Department.DNumber\}} Department)$$

Pronunciation: “The projection of Roll Number, Name, and Dname of Student left outer join on Student with Department with the condition Student dot Did equals Department dot DNumber"

Relational Algebra for Right Outer Join

```
SELECT Student.RollNumber, Student.Name , Department.Dname
FROM Student
RIGHT OUTER JOIN Department ON  Student.Did= Department.DNumber;
```

RollNumber	Name	Dname
S1	Ahmad	FOIT
S2	Aliya	FOIT
S3	Bushra	FOEE
S4	Bilawal	FOEE
NULL	NULL	FOSS

Assuming you have two tables A and B, and you want to perform a RIGHT OUTER JOIN on a condition C:

$$\pi_{RollNumber, Name, Dname}(Student \bowtie_{Student.Did = Department.DNumber} Department)$$

Pronunciation: “The projection of Roll Number, Name, and Dname of Student Right outer join on Student with Department with the condition Student dot Did equals Department dot DNumber"

Relational Algebra for Full Outer Join

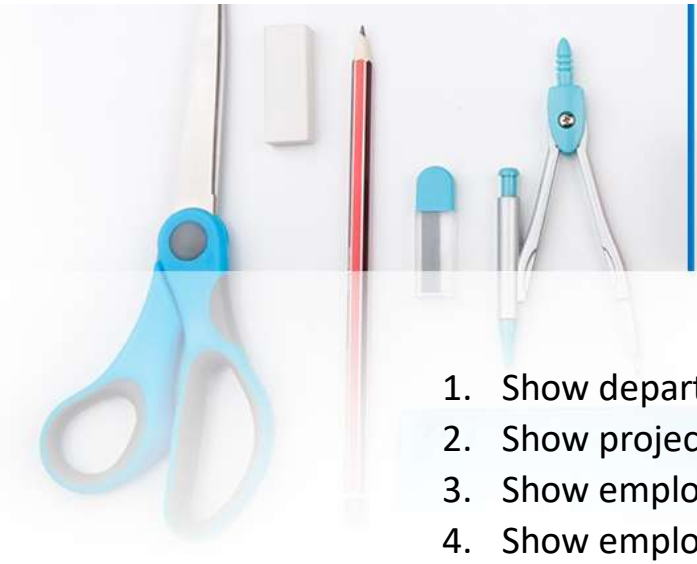
```
SELECT Student.RollNumber, Student.Name , Department.Dname
FROM Student
FULL OUTER JOIN Department ON  Student.Did= Department.DNumber;
```

RollNumber	Name	Dname
S1	Ahmad	FOIT
S2	Aliya	FOIT
S3	Bushra	FOEE
S4	Bilawal	FOEE
S5	Sameena	NULL
S6	Seher	NULL
NULL	NULL	FOSS

Assuming you have two tables A and B, and you want to perform a RIGHT OUTER JOIN on a condition C:

$$\pi_{RollNumber, Name, Dname}(Student \bowtie_{Student.Did = Department.DNumber} Department)$$

Pronunciation: “The projection of Roll Number, Name, and Dname of Student Full Outer Join on Student with Department with the condition Student dot Did equals Department dot DNumber"



Practice

1. Show department name with its manager's name
2. Show project name with its department name.
3. Show employee name with its dependent name.
4. Show employee name with the name of project he is working on.
5. Show employee name with its supervisor name.
6. Display name of manager and the date he starting managing the department.
7. Show department name with its department location.
8. Name the employee who works on a project that is located in 'Stafford'.
9. Name the employees who work on the project that is controlled by dept 5 or he manages dept 5.
10. Name all employees who have a dependents with the same first name as theirs.

SET OPERATIONS

- SQL has directly incorporated some set operations
- There is a union operation (**UNION)**, and in *some versions* of SQL there are set difference (**MINUS)** and intersection (**INTERSECT)** operations
- The resulting relations of these set operations are sets of tuples; *duplicate tuples are eliminated from the result*
- The set operations apply only to *union compatible relations* ; the two relations must have the same attributes and the attributes must appear in the same order

Thank You all!