SQL Joins

Inner Join

Left Join

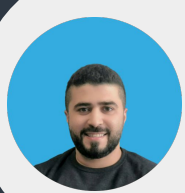
Right Join

Full Outer Join

Cross Join

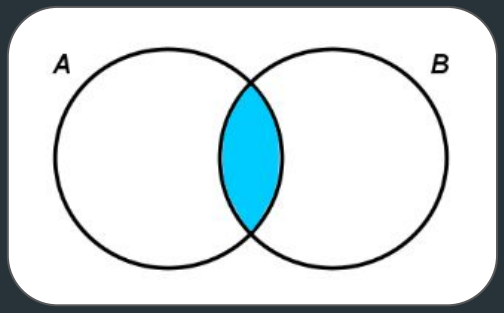
Self Join

Anti Join



Keivan Damirchi

Let's go



Inner Join

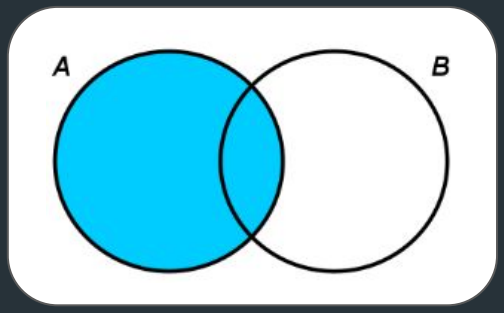
An inner join returns only the rows that have matching values in both tables. It is the most common type of join.

The time complexity of an inner join is O(n * m), where n and m are the number of rows in each table.

```
SELECT Customers.name, Orders.order_date
FROM Customers
INNER JOIN Orders
ON Customers.customer_id = Orders.customer_id;
```

Suppose we have two tables, Customers and Orders. Customers has columns for customer ID, name, and address, while Orders has columns for order ID, customer ID, and order date. To get a list of customers who have placed orders, we can use the this SQL statement.

Left Join



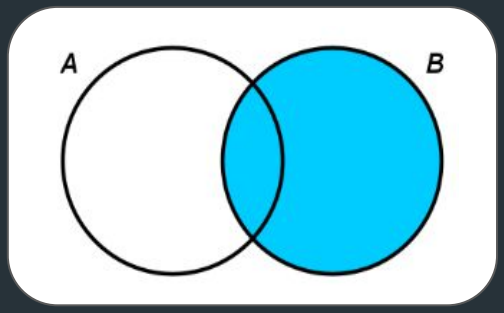
A left join returns all rows from the left table and the matching rows from the right table. If there is no matching row in the right table, the result will contain NULL values.

The time complexity of a left join is also O(n * m)

```
SELECT Customers.name, Orders.order_date
FROM Customers
LEFT JOIN Orders
ON Customers.customer_id = Orders.customer_id;
```

Using the same Customers and Orders tables as before, we can use a left join to get a list of all customers and their order dates (if they have placed an order).

Right Join



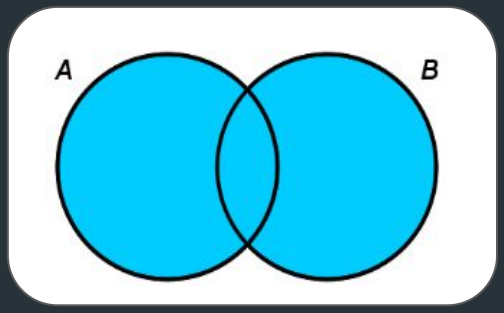
A right join is the opposite of a left join. It returns all rows from the right table and the matching rows from the left table. If there is no matching row in the left table, the result will contain NULL values.

The time complexity of a right join is also O(n * m)

```
SELECT Customers.name, Orders.order_date
FROM Customers
RIGHT JOIN Orders
ON Customers.customer_id = Orders.customer_id;
```

To get a list of all orders and the names of the customers who placed them (if known), we can use a right join.

Full Outer Join



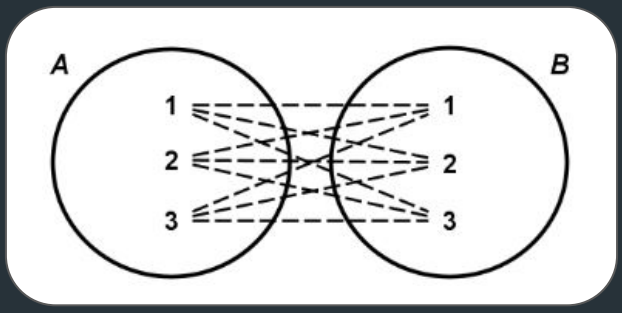
A full outer join returns all rows from both tables, including the ones that do not have a matching value in the other table. If there is no matching row, the result will contain NULL values.

The time complexity of a full outer join is also O(n * m)

```
SELECT Customers.name, Orders.order_date
FROM Customers
RIGHT JOIN Orders
ON Customers.customer_id = Orders.customer_id;
```

To get a list of all customers and all orders (if any), we can use a full outer join

Cross Join



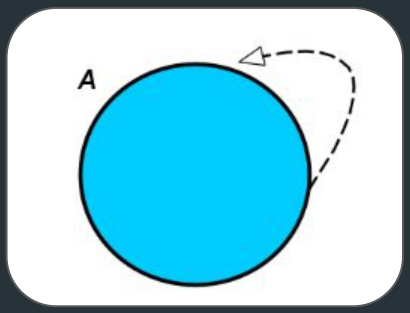
A cross join, also known as a cartesian join, returns the Cartesian product of the two tables. In other words, it returns all possible combinations of rows from both tables.

The time complexity of a cross join is also O(n * m)

```
SELECT Products.name, Colors.color_name
FROM Products
CROSS JOIN Colors;
```

Suppose we have two tables, Products and Colors. Products has columns for product ID, name, and price, while Colors has columns for color ID and color name. To get a list of all possible combinations of products and colors, we can use a cross join

Self Join



A self join is a join that is performed on a single table. It is used when we want to compare rows within the same table.

The time complexity of a self join is also O(n * m)

```
SELECT e.name AS employee_name, m.name AS manager_name
FROM Employees e
INNER JOIN Employees m
ON e.manager_id = m.employee_id;
```

Suppose we have a table called Employees, which has columns for employee ID, name, and manager ID. To get a list of all employees and their managers' names, we can use a self join

Anti Join

An anti join returns all rows from one table that do not have a matching row in the other table. It is also known as an exclusion join.

The time complexity of a anti join is also O(n * m)

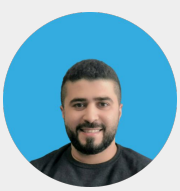
```
SELECT Customers.name
FROM Customers
LEFT JOIN Orders
ON Customers.customer_id = Orders.customer_id
WHERE Orders.order_id IS NULL;
```

Suppose we have two tables, Customers and Orders. Customers has columns for customer ID, name, and address, while Orders has columns for order ID, customer ID, and order date. To get a list of customers who have not placed any orders, we can use an anti join

Attention

It's worth noting that the time complexity of a join depends on the size of the tables being joined and the indexes available to the database. If there are indexes on the columns being joined, the time complexity can be improved. In general, it's a good idea to use indexes on columns that are frequently used in joins to improve performance.

Close();



Keivan Damirchi