Mentorness Internship task 1

 ARTICLE WRITING

Sai\_Vyshnavi\_Gudipalli\_MIP-DA-04\_SQL



Understanding SQL Joins: Natural, Cartesian, Inner, Outer, Left, and Right.

By Sai Vyshnavi Gudipalli

**INTRODUCTION**

SQL stands for structured query language. SQL joins are used for joining rows from two or more tables depending on the common factor or field from those tables.

For instance, let us look at this example where there are two tables movies and theaters.

|  |
| --- |
| Movies |
| Movie\_ID  Movie\_Name  Movie\_Language  Movie\_Rating |

|  |
| --- |
| Theaters |
| Movie\_ID  Theater\_Area  Theater\_Name  Screen\_Numbers |

Now we must find the movie that is being played in Screen 1, so for the solution of this problem we need to join the tables, the reason is as the screen number is present in theater table as you can see in the theater column and the name of the movie is present in the movie table.

Therefore by using the join statement, we can join the tables by using the common field that is the Movie\_ID and find the answer for the movie that is being played in screen 1.

This is how joins statements in SQL helps us to find solutions to these problems.

**TYPES OF JOINS**

There are mainly five types of joins, they are:

* Natural Join
* Cartesian Join
* Inner Join
* Left Outer Join
* Right Outer Join
* Full Outer Join
* Self Join

Now let us talk about each one in detail.

Now let us say we have 2 tables movies and theaters

The Table “Movies”

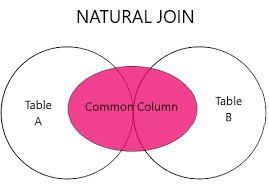
|  |  |  |  |
| --- | --- | --- | --- |
| **Movie\_ID** | **Movie\_Name** | **Movie\_Language** | **Movie\_Rating** |
| 1 | Damsel | English | 7 |
| 2 | Gami | Telugu | 6 |
| 3 | Bahubali | Telugu | 9 |
| 4 | Pushpa | Hindi | 8 |
| 5 | RRR | Hindi | 9 |

The Table “Theaters”

|  |  |  |  |
| --- | --- | --- | --- |
| **Movie\_ID** | **Theater\_ Area** | **Theater\_Name** | **Screen\_Numbers** |
| 1 | Gajuwaka | Melody | 1 |
| 3 | R.K.Beach | Inox | 2 |
| 5 | Seethamdhara | Imax | 3 |
| 6 | MVP Colony | Chitralaya | 4 |
| 7 | Jagadamba | Jagadamba | 5 |

**Natural Join**

In Natural join, only the common columns from table1(movies) and only the common columns for table2(Theaters) are shown in the result table. If the condition, if there is one common column in both the tables and they are of same data type condition is satisfied then the natural join can be done.



**Syntax for joining tables using Natural join:**

SELECT \*

FROM Movies

NATURAL JOIN Theaters;

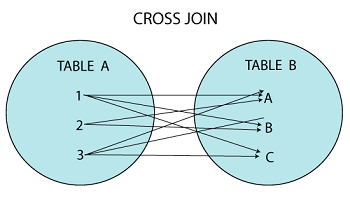
Now the result table of Natural join will be

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Movie\_ID** | **Movie\_Name** | **Movie\_Language** | **Movie\_Rating** | **Theater\_ Area** | **Theater\_Name** | **Screen\_Numbers** |
| 1 | Damsel | English | 7 | Gajuwaka | Melody | 1 |
| 3 | Bahubali | Telugu | 9 | R.K.Beach | Inox | 2 |
| 5 | RRR | Hindi | 9 | Seethamdhara | Imax | 3 |

So, The natural Join was performed taking the common column as Movie\_ID which has same name and same data type in both the tables i.e, is in movies table and Theaters table. And the tables are joined.

**Cartesian Join**

Cartesian join is also called cross join. It returns the product of both the tables, that means in the above example every movies table row is combined with every row in Theaters table. For example the movies tables contains 5 rows and the theaters table contains 5 rows, the resultant table of the Cartesian join will be 5\*5=25. So, the resultant table will have 25 rows. Only if WHERE condition is not used. If the WHERE condition is used it works similar to inner join, lets discuss more about it while we are talking about INNER JOIN.



**Syntax for joining tables using Cartesian join:**

SELECT Movies.Movie\_ID, Movies.Movie\_Name, Theaters.Theater\_Name ,Theaters. Screen\_Numbers

FROM Movies

CROSS JOIN Theaters;

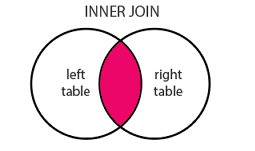
The resultant table of cross join is:

|  |  |  |  |
| --- | --- | --- | --- |
| Movie\_ID | Movie\_Name | Theater\_Name | Screen\_Numbers |
| 1 | Damsel | Melody | 1 |
| 1 | Damsel | Inox | 2 |
| 1 | Damsel | Imax | 3 |
| 1 | Damsel | Chitralaya | 4 |
| 1 | Damsel | Jagadamba | 5 |
| 2 | Gami | Melody | 1 |
| 2 | Gami | Inox | 2 |
| 2 | Gami | Imax | 3 |
| 2 | Gami | Chitralaya | 4 |
| 2 | Gami | Jagadamba | 5 |
| 3 | Bahubali | Melody | 1 |
| 3 | Bahubali | Inox | 2 |
| 3 | Bahubali | Imax | 3 |
| 3 | Bahubali | Chitralaya | 4 |
| 3 | Bahubali | Jagadamba | 5 |
| 4 | Pushpa | Melody | 1 |
| 4 | Pushpa | Inox | 2 |
| 4 | Pushpa | Imax | 3 |
| 4 | Pushpa | Chitralaya | 4 |
| 4 | Pushpa | Jagadamba | 5 |
| 5 | RRR | Melody | 1 |
| 5 | RRR | Inox | 2 |
| 5 | RRR | Imax | 3 |
| 5 | RRR | Chitralaya | 4 |
| 5 | RRR | Jagadamba | 5 |

Now from this table we can see that the resultant table of cross join consists of 25 rows as the Movies table and Theaters table contains 5 rows each.

**INNER JOIN :**

The inner Join returns only the common rows from both the tables based on the common column. So the table 1 compares each and every row of table 1 with each and every row of table 2, which ever rows are being matched only that will be present in the resultant table.



**Syntax for joining tables using Inner join:**

SELECT \*

FROM Movies

INNER JOIN Theaters

ON Movies.Movie\_ID = Theaters.Movie\_ID;

So the last line has the condition after “ON” statement, without that condition this syntax will become a cross join. And we are joining these two tables based on that common column that is the movie ID.

The inner Join and the Natural join are almost similar but the only difference is the inner join contains the common column more than once, where as in Natural Join , the common column occurs only once.

The resultant table of inner join is :

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Movie\_ID** | **Movie\_Name** | **Movie\_Language** | **Movie\_Rating** | **Movie\_ID** | **Theater\_ Area** | **Theater\_Name** | **Screen\_Numbers** |
| 1 | Damsel | English | 7 | 1 | Gajuwaka | Melody | 1 |
| 3 | Bahubali | Telugu | 9 | 3 | R.K.Beach | Inox | 2 |
| 5 | RRR | Hindi | 9 | 5 | Seethamdhara | Imax | 3 |

**LEFT OUTER JOIN**

The left outer join is also known as left join. The result table of left join contains all the rows from the left table i.e, the first table in our example the movies table and returns only the matching rows from the right table i.e, the second table in our example that is the Theaters table. After joining the missing values from the right table are represented by “NULL” values.



**Syntax for joining tables using Left Outer join:**

SELECT \*

FROM Movies

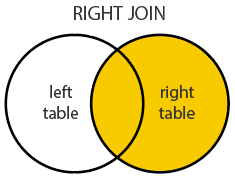
LEFT JOIN Theaters

ON Movies.Movie\_ID = Theaters.Movie\_ID;

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Movie\_ID** | **Movie\_Name** | **Movie\_Language** | **Movie\_Rating** | **Movie\_ID** | **Theater\_ Area** | **Theater\_Name** | **Screen\_**  **Numbers** |
| 1 | Damsel | English | 7 | 1 | Gajuwaka | Melody | 1 |
| 2 | Gami | Telugu | 6 | NULL | NULL | NULL | NULL |
| 3 | Bahubali | Telugu | 9 | 3 | R.K.Beach | Inox | 2 |
| 4 | Pushpa | Hindi | 8 | NULL | NULL | NULL | NULL |
| 5 | RRR | Hindi | 9 | 5 | Seethamdhara | Imax | 3 |

**RIGHT OUTER JOIN :**

Right outer join is also called as right join. The resultant table of right join returns all the rows from right table that is from the second table in our case all the rows from the theaters table are returned and only the matching rows from left table are returned that is from the first table in our example it is the movies table.



**Syntax for joining tables using Right Outer join:**

SELECT \*

FROM Movies

RIGHTJOIN Theaters

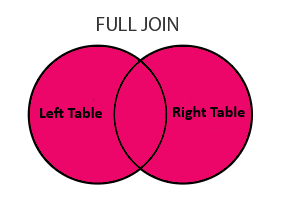
ON Movies.Movie\_ID = Theaters.Movie\_ID;

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Movie\_ID** | **Movie\_Name** | **Movie\_Language** | **Movie\_Rating** | **Movie\_ID** | **Theater\_ Area** | **Theater\_Name** | **Screen\_**  **Numbers** |
| 1 | Damsel | English | 7 | 1 | Gajuwaka | Melody | 1 |
| 3 | Bahubali | Telugu | 9 | 3 | R.K.Beach | Inox | 2 |
| 5 | RRR | Hindi | 9 | 5 | Seethamdhara | Imax | 3 |
| NULL | NULL | NULL | NULL | 6 | MVP Colony | Chitralaya | 4 |
| NULL | NULL | NULL | NULL | 7 | Jagadamba | Jagadamba | 5 |

**FULL OUTER JOIN :**

Full outer join is also known as full join. It returns all the rows from both the tables and makes sure no data is lost in between.

Suppose if there are any missing values in the rows that is represented by “NULL” value.



**Syntax for joining tables using Full Outer join:**

SELECT \*

FROM Movies

FULL JOIN Theaters

ON Movies.Movie\_ID = Theaters.Movie\_ID;

Now the resultant table of the full join will be:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Movie\_ID** | **Movie\_**  **Name** | **Movie\_**  **Language** | **Movie\_**  **Rating** | **Movie\_**  **ID** | **Theater\_ Area** | **Theater\_Name** | **Screen\_**  **Numbers** |
| 1 | Damsel | English | 7 | 1 | Gajuwaka | Melody | 1 |
| 2 | Gami | Telugu | 9 | NULL | NULL | NULL | NULL |
| 3 | Bahubali | Telugu | 9 | 3 | R.K.Beach | Inox | 2 |
| 4 | Pushpa | Hindi | 7 | NULL | NULL | NULL | NULL |
| 5 | RRR | Hindi | 9 | 5 | Seethamdhara | Imax | 3 |
| NULL | NULL | NULL | NULL | 6 | MVP Colony | Chitralaya | 4 |
| NULL | NULL | NULL | NULL | 7 | Jagadamba | Jagadamba | 5 |

**SELF JOIN:**

Self join is a join using which the table is joined to itself. This join is used to compare the columns of the same table. The self join can be used with any of the joins that we discussed above that is inner join, left join, right join, Cartesian join or full join.

A and B are aliases of table1.i.e, the Movies table,

Now let us say we want all the movie names that have same languages

**The Syntax for that will be**

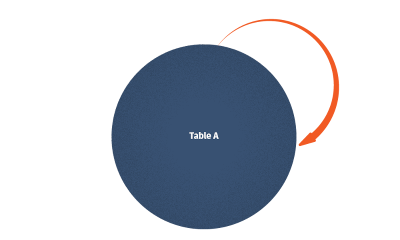
SELECT A. Movie\_Language, A. Movie\_Name , B. Movie\_Name

FROM Movies A

INNER JOIN Movies B

ON A. Movie\_Language = B. Movie\_Language

AND A. Movie\_Name > B.NAME;



Here we used AND clause so that the Movie\_name does not repeat twice in the table

|  |  |  |
| --- | --- | --- |
| Movie\_Language | Movie\_Name | Movie\_Name |
| Telugu | Gami | Bahubali |
| Hindi | Pushpa | RRR |

Now as we can see from the table we have all the movie names of the same languages.

**CONCLUSION:**

Hi guys we are at the end of the joins article, so in this article we have learned what are joins, how they work, how are they important to get the solutions of the given data. Using these sql joins we can also filter out the irrelevant data. Joins are very important part of sql as they make the complex problems very easy to solve. And sometimes also the incorrect usage of these joins make your query slower. That’s all about the sql joins.

Hope you like the article. Happy coding.

