Joins In DBMS

Joins can be simply defined as the combining or merging the related tuples from the two different relations into a single type. It can be said that it is similar to cartesian product except the fact that in cartesian product, we get all the possible combinations of relations while in join only those combinations can be formed that satisfies some matching conditions. A cartesian product is followed by a selection process results in join.

We can classify joins basically into two types

1. INNER JOINS: These joins are the one that has the tuples that satisfy some conditions and rest are discarded . Further they are classified as

- Theta join
- Equi join
- Natural join

2. OUTER JOINS: These have all the tuples from either or both the relations. Further they are classified as

- Left outer join
- Right outer join
- Full outer join

Let us now move on to the study the classified types with examples in detail.

a) **Theta join(θ)** - They have tuples from different relations if and only if they satisfy the theta condition, here the comparison operators (≤, ≥, <, >, =, ¬)come into picture. Let us consider simple example to understand in a much better way, suppose we want to buy a

mobile and a laptop, based on our budget we have thought of buying both such that mobile price should be less than that of laptop. Look at the tables below,

MOBILE

MODEL	PRICE
Asus	10k
Samsung	20k
Iphone	50k

LAPTOP

MODEL	PRICE
Acer	20k
HP	35k
Apple	80k

Now, we have considered the condition as the cost of the mobile should be less than that of laptop so our resulting table will have only those tuples that satisfy this condition.

AFTER JOIN

Asus	Acer
Asus	HP
Asus	Apple
Samsung	HP
Samsung	Apple
Iphone	Apple

b) **Equi join** - As the name itself indicates, if suppose the join uses only the equality operator then it is called as equi join.

c) **Natural join** - It does not utilize any of the comparison operator. Here the condition is that the attributes should have same name and domain. There has to be at least one common

attribute between between two relations. It forms the cartesian product of two arguments, performs selection forming equality on those attributes that appear in both relations and eliminates the duplicate attributes. Consider the example, where two tables namely employment table and department table have been shown. e

EMPLOYMENT

NAME	EMPID	DPT_NAME
A	11	Sales
B	12	Finance
C	13	Finance

DEPARTMENT

DPT_NAME	MANAGER
Finance	M1
Sales	M2

Looking at above tables we realize that they have a common attribute called DPT_NAME, thus after the natural join the table becomes as

Name	EMPID	DPT_NAME	MANAGER
A	11	Sales	M2
B	12	Finance	M1
C	13	Finance	M1

d) **Left outer join** - All the tuples of left table is displayed irrespective of whether it satisfies the matching conditions. Thus in the left all the tuples have been displayed but in the right only those are present that satisfy the matching conditions. For example consider below example of two tables - country table that has 3 records and state table that has 4 records.

Country names are given the country_id that has to match with the country_id in the state table. India's state is Karnataka and Tamil Nadu, state of Pakistan is Islamabad but Nepal does not have state in the given table so right part will be null.

COUNTRY

COUNTRY_ID	COUNT_NME
1	India
2	Pakistan
4	Nepal

STATE

STATE_ID	COUNTRY_ID	STATE_NME
1	1	Karnataka
2	1	Tamil Nadu
3	2	Islamabad
4	NULL	Bangladesh

So for left join, left side table have all the values but right side only has those whose COUNTRY_ID has been matched.

LEFT OUTER JOIN

COUNTRY_ID	COUNT_NME	STATE_ID	COUNTY_ID	STATE_NME
1	India	1	1	Karnataka
1	India	2	1	Tamil nadu
2	Pakistan	3	2	Islamabad
4	Nepal	NULL	NULL	NULL

Bangladesh has not occurred since there is no match found.

e) **Right outer join** - All the tuples of right table are displayed irrespective of whether it satisfies the matching conditions or not.. Thus in the right, all the tuples have been displayed but in the left only those are present that satisfy the matching conditions. The previous example can be implemented here as well.

RIGHT OUTER JOIN

COUNTRY_ID	COUNT_NME	STATE_ID	COUNTY_ID	STATE_NME
1	India	1	1	Karnataka
1	India	2	1	Tamil nadu
2	Pakistan	3	2	Islamabad
NULL	NULL	4	NULL	Bangladesh

f) **Full outer join**- Tuples from both the relations takes part irrespective of whether it has the matching or non-matching conditions. Example is as shown

FULL OUTER JOIN

COUNTRY_ID	COUNT_NME	STATE_ID	COUNTY_ID	STATE_NME
1	India	1	1	Karnataka
1	India	2	1	Tamil nadu
2	Pakistan	3	2	Islamabad
4	Nepal	NULL	NULL	NULL
NULL	NULL	4	NULL	Bangladesh