DBMS Lecture 5

Joins:

Basic: Till now whenever we do the read query, whenever we try to get some data from the table,We will get only one table from the database.

In scaler codebase, we have 2 tables, students and batches. **Q**- Find all the students in "April21 intermediate" batch.

Solution: a. Go to batches table and find the batch with the given name. b. Find the students with batch_id in students table.

In real system, there is going to be an application server and database. Application server uses network to talk to database. But network calls are going to be slower than CPU operations.

Now, almost 4 network transfers are there. Asuume that to send one data, it takes approx 250ms, to find all the students of a particular batch: 1sec. So it's slower.

When we were designing bad databases, Info about students and batches are in the same table. Problems: Redundancy.

Even though there is redundancy, but benefit is it will make the queries faster because all data is at single place. *NoSQL databases kind of go with denormalization.*

==By using joins we can get kind of all the data at same place without actually denormalizing the schema.==

We have students table and batches table. Joins: Merging two things. ==Merging/Joining 2 tables without actually joining them(join it for the query, not in reality).==

Students Join Batches *Bringing the rows of both tables together*

How to combine two rows, like the first row of student will combine to same batch_id.

Now we have join the tables. But the question is, **How to construct this table??**

Types of Join:

1. Inner Join:

Combine rows of both the tables based on the given condition. In an Inner Join we have two rows, Example: There are two databases students and batches, whenever we do inner join between them, there is always a condition that happens.

```
 select *
from students
inner join batches
on students.batch_id=batches.id
where batches.name="Apr21";
```

Let's say there are 3 students A,B,C and there are 2 batches,

for every row of student: for every row of batches: if the join condition is satisfied: put a merged row in the answer.

Now, 1 data of batch be present multiple times. If 1 student belong to multiple batches, they will also have data twice because for that row of the student it will match against 2 rows of the batches. That means that data of the student is present twice.

In students table, **Q: Print all the students with all their phone numbers.**

```
 select * from students
inner join phone-numbers
on students.id = phone.id;
```

Inner joins are so common, this inner keyword is optional.

==Join=Inner Join==

In MYSQL Workbench, In sql_store database, **Q**- For every order get the order_id, customer_id, status, shipped_date, customer first_name and phone_number.

```
 select
from orders
join customers
on orders.customer_id=customers.customer_id;
```

Here customer_id is present twice. ==The final number of columns= sum of number of columns on both the tables.==

```
 select orders.order_id,orders.customer_id, orders.status,orders.shipped_date, customers.first_name, cu
from orders
join customers
on orders.customer_id=customers.customer_id;
```

We can name the tables as well.

```
 select o.order_id,o.customer_id, o.status,o.shipped_date, c.first_name, c.phone_number
from orders o
join customers c
on o.customer_id=c.customer_id;
```

It reduce the size of query.

Now, here the condition was, `orders.customer_id= customers. id`

What if one order has customer_id as NULL. It will not be the part of final answer because NULL is not equal to anything.

---

2. Self Join:

Scaler introduces a new feature where every student is assigned a peer reviewer who are also students. So they are also going to have students_id.

Q: Print the name of every student with the name of their peer. Expected output: Assume peer_review_id is a seperate table.

We'll do a simple inner join between these 2 tables and get the answer.

```
 select *
from students s
join peer_review p
on s.peer_id=p.id;
```

Only difference between real and this scenerio is: the peer_review is not a seperate table.

```
 select *
from students s
join students p
on s.peer_id=p.id;
```

==A self join is nothing but a inner join with the same table.==

In a self join we can't avoid creating aliases because in On condition How can SQL know which table to join with which table.

In sql_hr, MYSQL Workbench, Table employees, reports_to column is the employee_id of their manager.

Q- Print the name of every employee with the name of their manager.

```
 Use sql_hr;

Select e.first_name,m.first_name
from employees e
join employees m
on e.reports_to=m.employee_id;
```

Till now we have done joins in only 1 condition. But we can do joins based on multiple conditions and this is known as ==Compound Join.==

```
 select *
from table A
join table B
on {condition1}
and {condition2}
```

Joining Multiple tables:

In scaler codebase: Students table, batches table and Instructors table.

Q- Print student name, instructor name for every student. Expected Output: We Can't directly get the instructor from students. No mapping from student to instructor.

So, **Create a Middle table**,with: *id,name,batchid,b. id,b. instid*

This table when combined with instructor table we can get the information we want.

First, Join between students and batches and then combined thing with instructors table.

```
 select s.name,i.name
from students s
join batches b
on s.batch_id=b.id
join instructors i
on b.inst_id=i.id;
```

Just combine them 1 after the other.

In MYSQL Workbench, In sqlstore, **Q**- For every order, Print order_id,order_date, product_name and quantity bought.

All this information should be in 1 table only.

Some info is there in orders table, some in order_items table, and some in products table. So, we need to do a join between these.

From orders, we know the order_id and probabaly we can combine that with the order_items and here we get the product_id which we will combine with products to get the final names.

```
 Use sql_store;


select o.order_id,o.order_date,p.name,oi.quantity
from orders o
join order_items oi
on o.order_id=oi.order_id
join products p
on oi.product_id=p.product_id;
```

If we do **A join B**, If a particular row of the left side didn't match the condition for any row of B, then that will not be returned. *for every row of A(row1) for every row of B(row2) if row1 and row2 satisfy conditions then put them together.*

==A particular row of A might not be present in the join if it doesn't match any condition.== ==A particular row of B might not be present in the join if it doesn't match any condition== But we don't want this. If the particular condition doesn't satisfy it should still be there.

==In case a row doesn't satisfy any condition it should be returned in join with NULL on other side.==

We have 2 tables, students and batches

Q- Print the name of every student with the name of their batch, if a student is not present in any batch, print their name with NULL as the batch_name.

Inner join will not work here because they satisfy conditions.

Outer join:

Outer Joins will also try to satisfy the condition but if the condition is not satisfied, they will also returned the rows of the other side.

1. Left Join
2. Right Join
3. Full Join

Left Join:

==If a row of left side doesn't match with any row of the right side, it will be added in final answer with NULL.== If we do a left join between these 2 tables, final answer is going to have 5 columns.

All the rows of left side will be returned.

Right Join:

==All rows of right side will be returned.==

Full Join: ==If any row of any side is not present, Put that with a NULL.==

```
 Use sql_hr;

Select e.first_name,m.first_name
from employees e
left join employees m
on e.reports_to=m.employee_id;
```

Difference is: Whenever there is a employee who doesn't have a manager so they also be returned with a NULL.

```
 Use sql_hr;

Select e.first_name,m.first_name
from employees e
right join employees m
on e.reports_to=m.employee_id;
```

If some person isn't have any reportees they will be returned, first_name would be NULL in this case.

==MYSQL doesn't support FULL JOIN.==

Now,

```
 select *
from students s
join batches b USING(batch_id);
```

Works if batch_id is present on both the tables and we want to do a join on s.batch_id=b.batch_id. Just reducing the amount of lines in code.

We can also put 2 columns in USING clause.

NATURAL JOIN:

No need to write any condition. No ON and USING. Automatically join based on columns with same name.

```
 select o.order_id,o.customer_id, o.status,o.shipped_date, c.first_name, c.phone_number
from orders o
natural join customers c;
```

It goes through both the tables, find the column with the same name on both the tables, tries to do the join based on that column.

CROSS JOIN:

No condition. Join every row of left side with every row on right side. If students has 9 rows and batches has 10 rows, The cross join would have (9*10)=90 rows. It is needed when we want every row from both the tables.

If we write a condition in cross join, it becomes INNER JOIN. Because a cross join will never going to give a row with NULL on the other side, it is going to give a combination with all the available rows from other side but not NULL.

==CROSS JOINS are slower than INNER JOINS because they are practically going from every row of A to every row of B.==
