SCALER

Joins - 2

TABLE OF CONTENTS

12th Hard day challenge :

1. Assignments + Revision
2. Backlog (Assignments of prev. session)
3. Additional Questions

Compound Joins

- Compound Joins are similar to regular joins.
- We just apply multiple conditions to multiple columns.

Example

(Between range)

For every film, name films which were released 2 years before current film and two years after current film. and rental rate > rental rate of current film.

f1

Films

name	release_year	rate
Hera Pheri	2008	2
Robot	2009	3
Welcome	2011	4
Bahubali	2016	2

2006
2010

f2

Films

name	release_year	rate
Hera Pheri	2008	2
Robot	2009	3
Welcome	2011	4
Bahubali	2016	2

Films

name	release_year	rate	name	release_year	rate
Hera Pheri	2008	2	Hera Pheri	2008	2
Hera Pheri	2008	2	Robot	2009	3
Robot	2009	3	Hera Pheri	2008	4
Robot	2009	3	Robot	2009	3
Robot	2009	3	Welcome	2011	4
Welcome	2011	4	Robot	2009	3
Welcome	2011	4	Welcome	2011	4
Bahubali	2016	2	Bahubali	2016	2

```
select   f1.name , f2.name
from films f1
join films f2
on ( f2.release_year between f1.release_year - 2 and f1.release_year + 2 )
    and ( f2.rental_rate > f1.rental_rate ) ;
```

Types of Joins

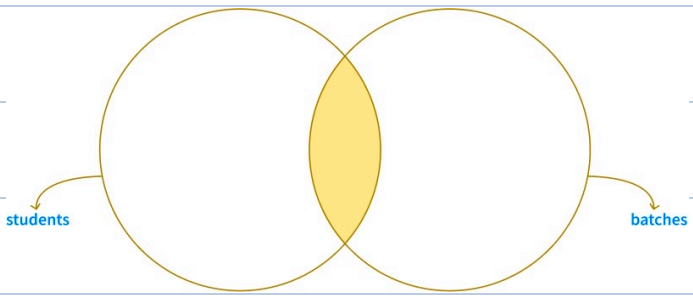
1. **Inner Join**
2. **Outer Join**
3. **Full Join**

< / > Syntax

```
SELECT  *
FROM students s
JOIN   batches b
ON     s.b_id = b.id
```

1. Inner Join

The join where we will get data for matching conditions only, is known as **Inner Join**.



2. Outer Join

It includes all the rows even if the condition doesn't match.

Types of Outer Join :

a. Left Join

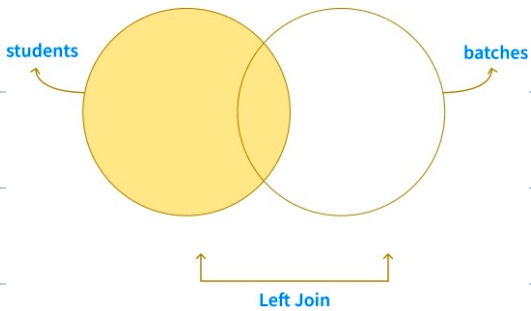
b. Right Join

c. Full Join → Not available in MySQL

a) Left Join

- Gives all the rows from left table and only matching rows from the right table.

Example :



Get all students along with their batch_name.

Also give the data for unassigned students.

Students

id	name	b_id	psp
1	John	1	80
2	Jane	Null	90
3	Jim	2	85
4	Jenny	3	95
5	Jack	2	78

Batches

b_id	name
1	A
2	B
3	C
4	D

left join

```
select *
from students s
left join batches b
on s.b_id = b.b_id ;
```

Output

Students_table | batch_table

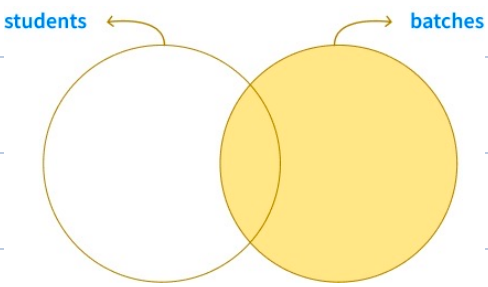
id	name	b_id	psp	b_id	name
1	John	1	80	1	A
2	Jane	Null	90	Null	Null
3	Jim	2	85	2	B
4	Jenny	3	95	3	C
5	Jack	2	78	2	B

b) Right Join

- Gives all the rows from right table and only matching data from the left table.

Example :

Get all students with their assigned batches. And get all the batch names even though no students are assigned to them.



Students

id	name	b_id	psp
1	John	1	80
2	Jane	Null	90
3	Jim	2	85
4	Jenny	3	95
5	Jack	2	78

Batches

b_id	name
1	A
2	B
3	C
4	D

Right join

```
select *
from students s
right join batches b
on s.b_id = b.b_id;
```

Output

Students_table | batch_table

id	name	b_id	psp	b_id	name
1	John	1	80	1	A
5	Jack	2	78	2	B
3	Jim	2	85	2	B
4	Jenny	3	95	3	C
Null	Null	Null	Null	4	D

c) Full Join

- We will get all rows from left as well as right table, while putting null values in corresponding tables for which value doesn't match.

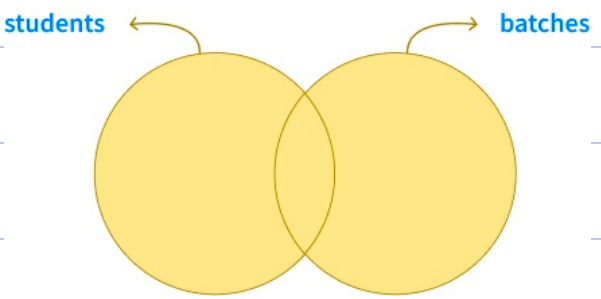
Example :

Students

id	name	b_id	psp
1	John	1	80
2	Jane	Null	90
3	Jim	2	85
4	Jenny	3	95
5	Jack	2	78

Batches

b_id	name
1	A
2	B
3	C
4	D



Output

Students_table | batch_table

id	name	b_id	psp	b_id	name
1	John	1	80	1	A
2	Jane	Null	90	Null	Null
3	Jim	2	85	2	B
4	Jenny	3	95	3	C
5	Jack	2	78	2	B
Null	Null	Null	Null	4	D

3. Cross Join

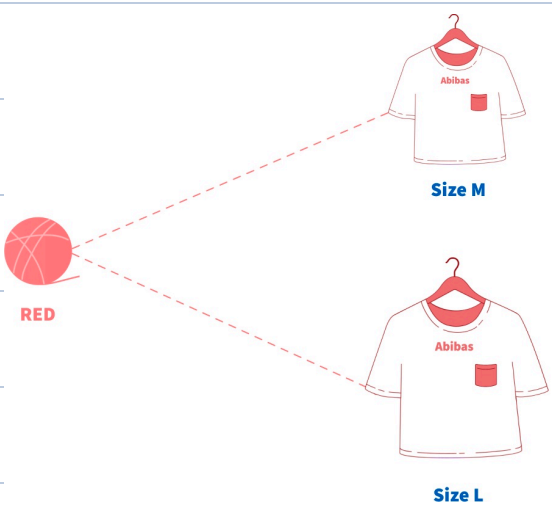
- Imagine you work with an international fashion brand. They manufacture only limited stock of apparels. Now they aim to release shirts in various sizes, offering both red and purple color options for each size.
- From the provided tables listing sizes and colors, we need to generate all possible combinations of sizes and colors.

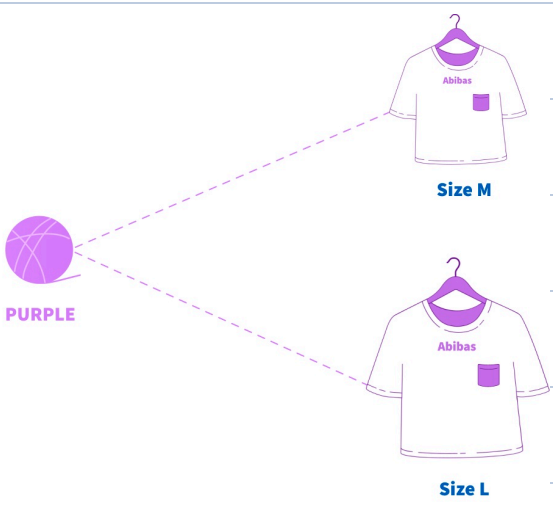
Colors

id	name
1	Red
2	Purple

Sizes

id	name
1	M
2	L





< / > *Syntax*

```
SELECT *
FROM colors
CROSS JOIN sizes;
```

Final Table

id	name	id	size
1	Red	1	M
1	Red	2	L
2	Purple	1	M
2	Purple	2	L

Size of output = $m \times n$

4. Natural Join & Natural Joins using clause

- Joins two tables based on equality of columns which are common in both the tables.

  It will check equality of every common column.

Students

id	name	b_id	psp
1	John	1	80
2	Jane	1	90
3	Jim	2	85
4	Jenny	3	95
5	Jack	2	78

Batches

b_id	b_name
1	A
2	B
3	C

Students

id	name	b_id	psp
1	John	1	80
2	Jane	1	90
3	Jim	2	85
4	Jenny	3	95
5	Jack	2	78

Batches

b_id	b_name
1	A
2	B
3	C

Output

Students_table

batch_table

id	name	b_id	psp	b_id	b_name
1	John	1	80	1	A
2	Jane	1	90	1	A
3	Jim	2	85	2	B
4	Jenny	3	95	3	C
5	Jack	2	78	2	B

On vs Where

→ When we write 'join using 'where' clause, it is not going to participate in 'join hence internally it's a cross 'join which is filtered by 'where' clause condtn, where is inefficient.

→ Use 'ON' clause to 'join tables.

pseudo Code:

ON Clause

```
ans = []
for row1 in table1:
    for row2 in table2:
        if ('ON' condtn).matches:
            ans.add(row1 + row2)

for row in ans:
    print(row)
```

where clause

```
ans = []
for row1 in table1:
    for row2 in table2:                    # cross join
        ans.add(row1 + row2)

for row in ans:
    if condtn(where clause).matches:
        print(row)
```

Union

< *Question* > : Given data of Scaler's database, get name of everyone associated with Scaler.

Students

id	name
1	Rahul
2	Satish
3	Mohit

~~Students~~ Instructors

id	name
1	Naman
2	Prateek

Investors

id	name
1	Rahul
2	Mohit

Output

name
Rahul
Mohit
Satish
Naman
Prateek

```
select name from students
union
select name from instructors
union
select name from invitors ;
```

* Union filters out duplicate value.
* Use Union all to get all the values.