Joins

Aoun-Haider

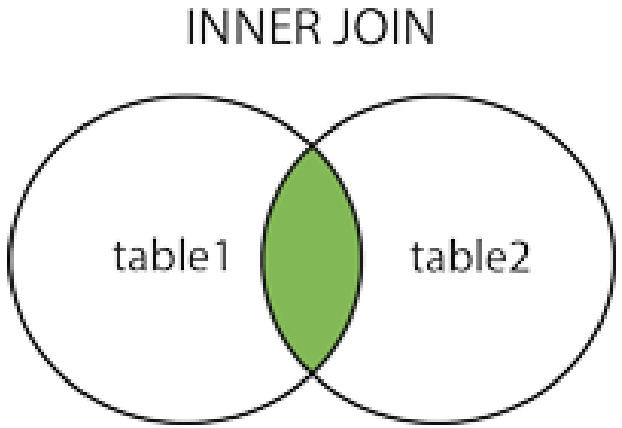
FA21-BSE-133@cuilahore.edu.pk

Types:

- Left join/ left outer join
- Right join/ right outer join
- Full join/ full outer join
- Inner join
- Natural join
- Semi-join
- Equi-join
- Theta join
- Anti-join
- Cross join
- Self join

Inner Join:

Returns records that have matching values in both tables



ID	Name	Salary
1	John	40000
2	Alex	25000
3	Simon	43000

+

ID	Age	Status
1	32	Married
3	26	Married

=

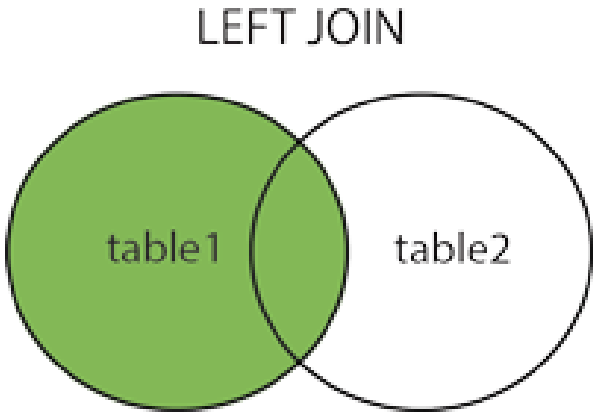
ID	Name	Age	Salary	Status
1	John	32	40000	Married
2	Alex	NULL	25000	NULL
3	Simon	26	43000	Married

Outer Join:

- Unlike inner join the outer join may contain the records that doesn’t satisfy the join condition along with the records that satisfy it. There are three types of outer join namely –
- Left Outer Join
- Right Outer Join
- Full Outer Join

Left Join:

Fetches all the matched entries
And all left table entries whether
They match or not.

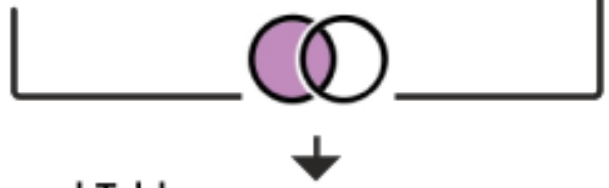


Left Table

Date	CountryID	Units
1/1/2020	1	40
1/2/2020	1	25
1/3/2020	3	30
1/4/2020	4	35

Right Table

ID	Country
1	USA
2	Canada
3	Panama

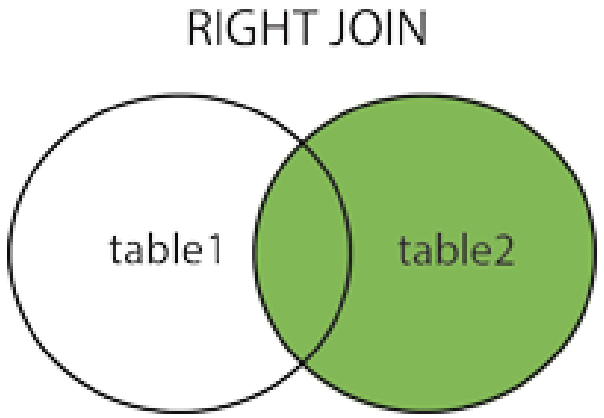


Merged Table

Date	CountryID	Units	Country
1/1/2020	1	40	USA
1/2/2020	1	25	USA
1/3/2020	3	30	Panama
1/4/2020	4	35	*null*

Right Join:

Fetches all the matched entries
And all right table entries whether
They match or not.



Left Table

Date	CountryID	Units
1/1/2020	1	40
1/2/2020	1	25
1/3/2020	3	30
1/4/2020	4	35

Right Table

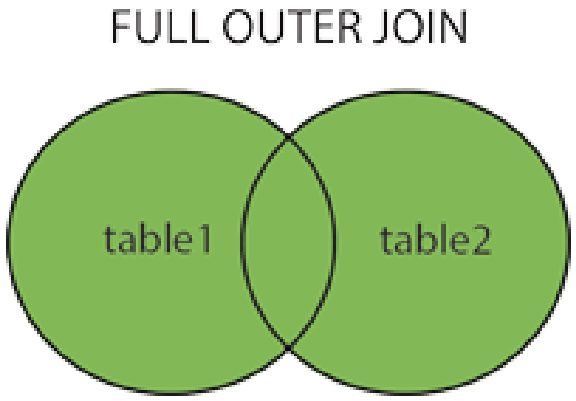
ID	Country
3	Panama

Merged Table

Date	CountryID	Units	Country
1/3/2020	3	30	Panama

Full outer join:

Fetches all the matched entries of both table. Non-matching entries will be assigned null.



Left Table

Date	CountryID	Units
1/1/2020	1	40
1/2/2020	1	25
1/3/2020	3	30
1/4/2020	2	35

Right Table

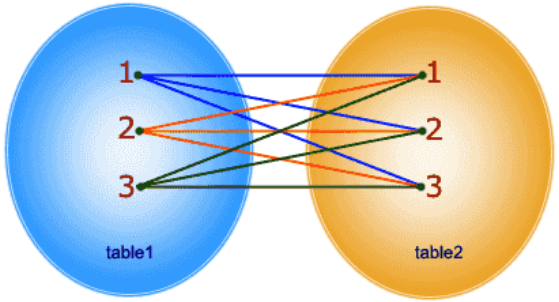
ID	Country
1	USA
2	Canada
3	Panama
4	Spain

Merged Table

Date	CountryID	Units	Country
1/1/2020	1	40	USA
1/2/2020	1	25	USA
1/4/2020	2	35	Canada
1/3/2020	3	30	Panama
null	*null*	*null*	Spain

Cross Join:

Same as cartesian product of two tables.




```
SELECT foods.item_name,foods.item_unit,
company.company_name,company.company_city
FROM foods
CROSS JOIN company;
```

Co.ID	Co.Name	Co.City
18	Order All	Boston
15	Jack Hill Ltd	London
16	Akas Foods	Delhi
17	Foodies.	London
19	sip-n-Bite.	New York

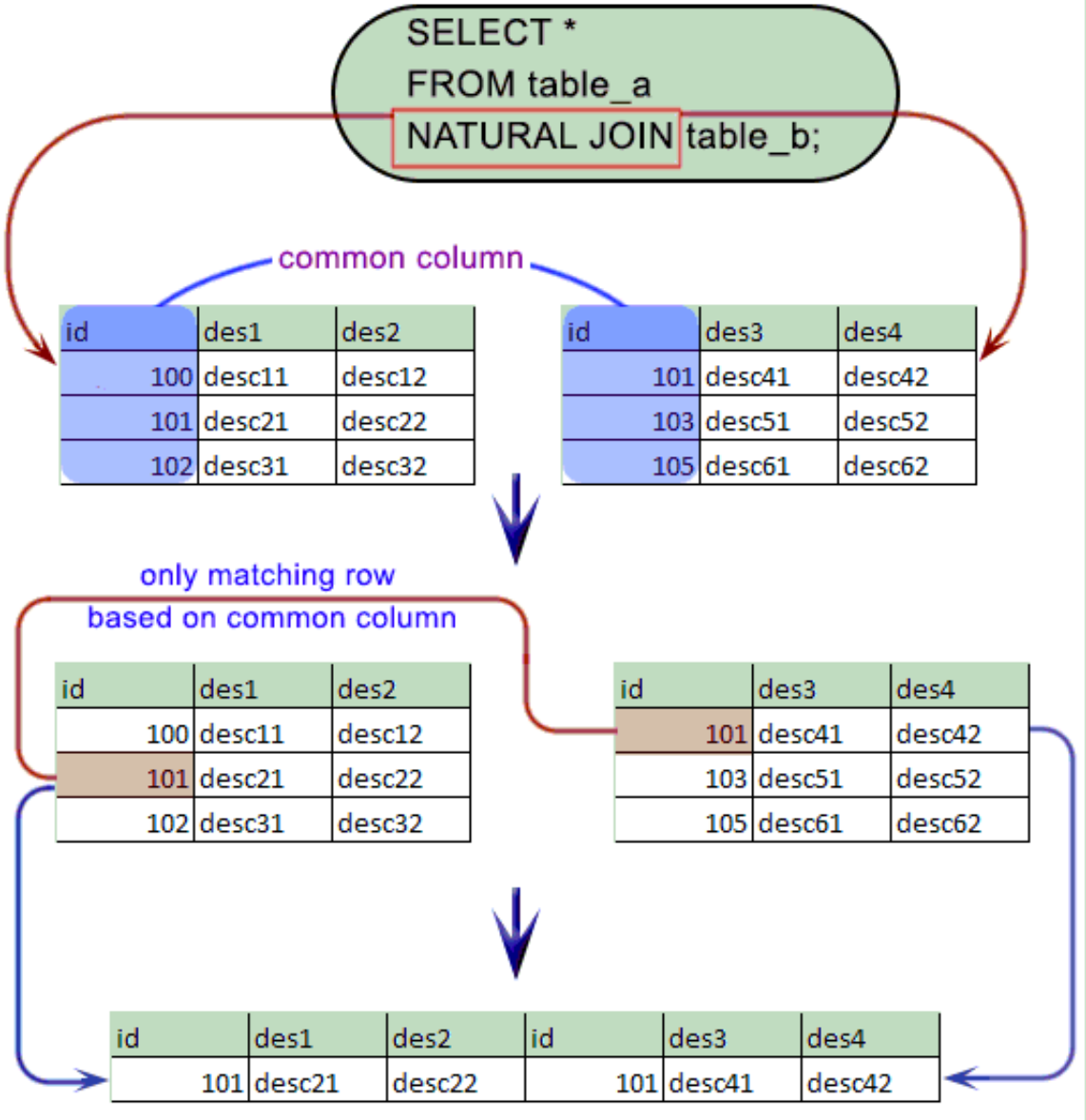
Company

I.ID	ITEM_NAME	I.Unit	Co.ID
1	Chex Mix	Pcs	16
6	Cheez-It	Pcs	15
2	BN Biscuit	Pcs	15
3	Mighty Munch	Pcs	17
4	Pot Rice	Pcs	15
5	Jaffa Cakes	Pcs	18
7	Salt n Shake	Pcs	-

Foods

Natural Join:

Returns only matching rows of both tables.



Semi Join:

```
SELECT DISTINCT s.id
FROM  students s
    LEFT JOIN grades g ON g.student_id = s.id
WHERE g.student_id IS NOT NULL
```

Now the same with left semi-join:

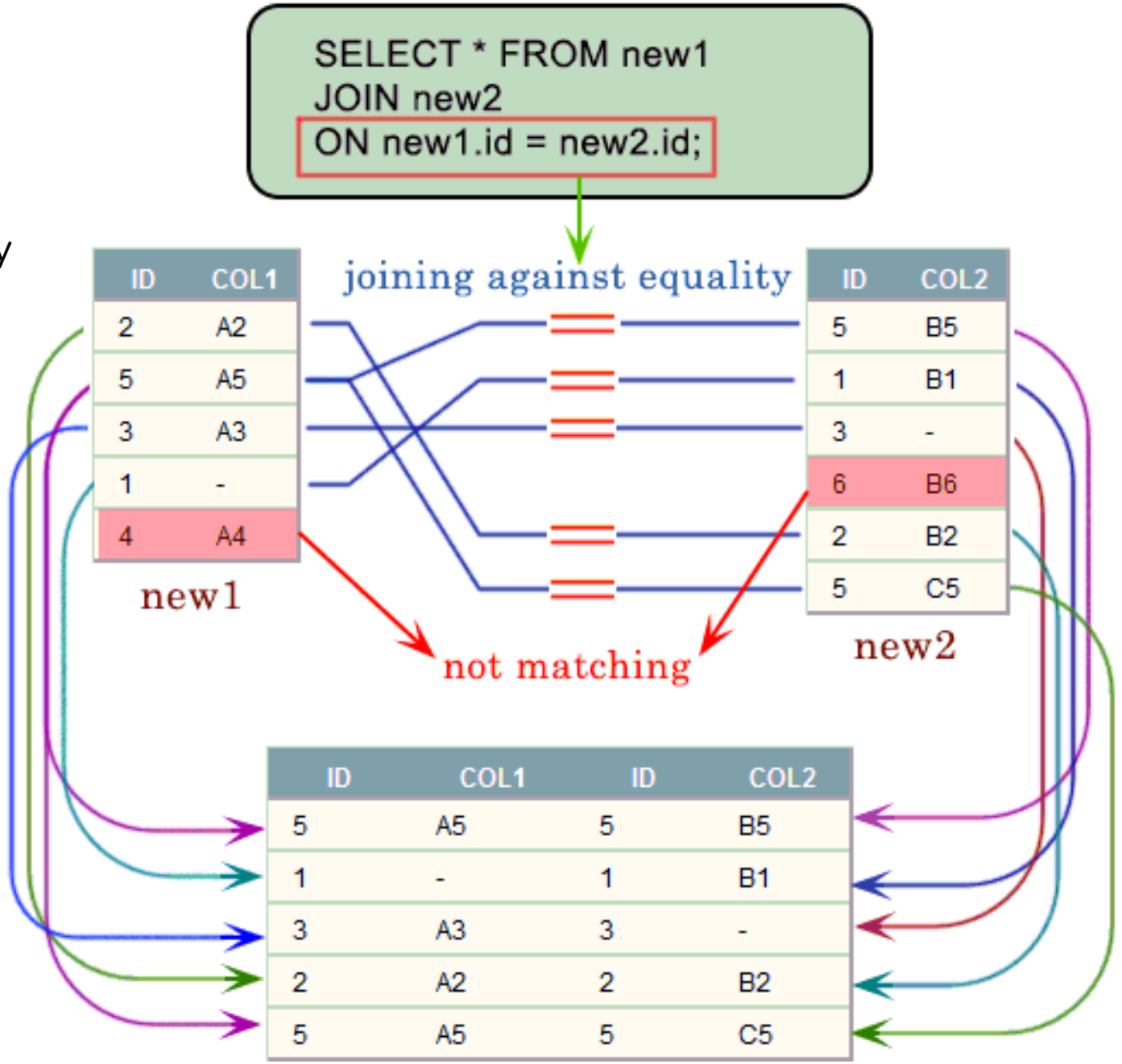
```
SELECT s.id
FROM  students s
WHERE EXISTS (SELECT 1 FROM grades g
            WHERE g.student_id = s.id)
```

Cont.

- Difference between regular join and semi-join is that it eliminates duplicate tuples and returns only 1st matching values. While regular join returns all the rows without taking care of duplicates.
- In most cases, semi-join uses EXISTS, IN, NOT IN, DISTINCT clause to perform required operation.

Equi join:

SQL EQUI JOIN performs a JOIN against equality or matching column(s) values of the associated tables. An equal sign (=) is used as comparison operator in the where clause to refer equality.



Self Join:

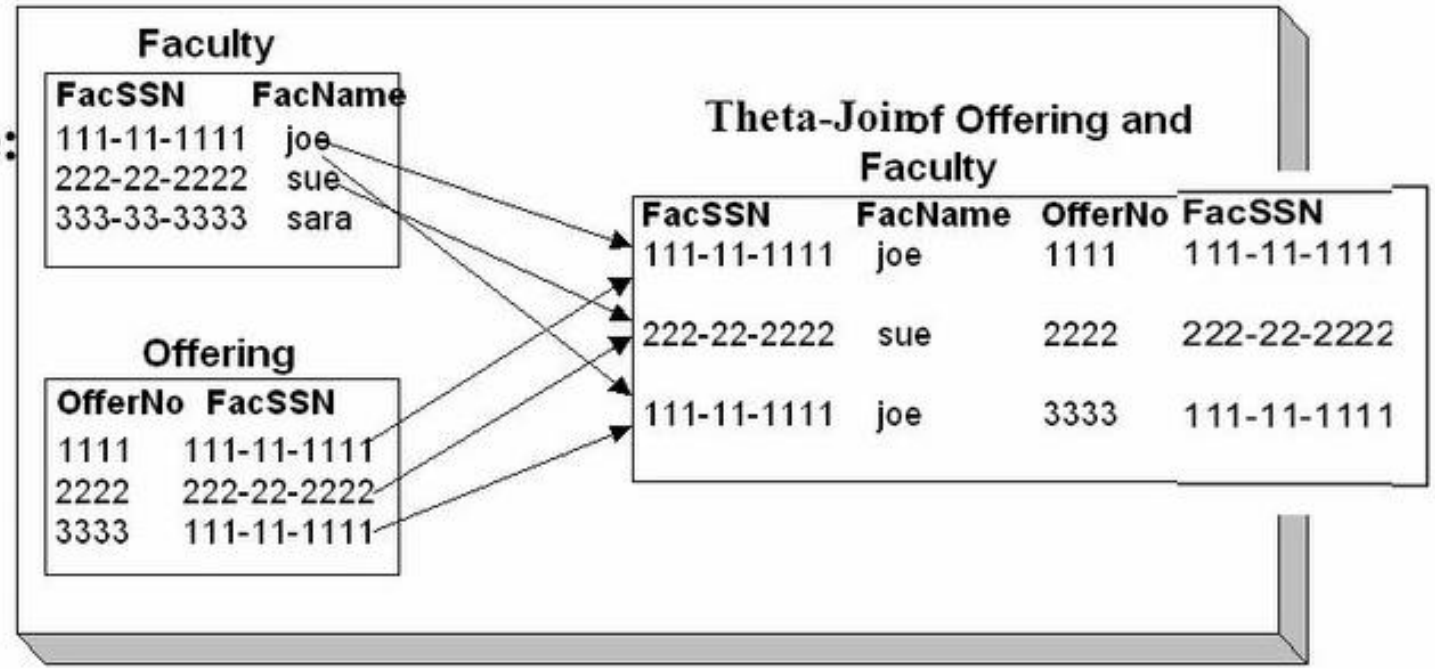
- A self join is a join in which a table is joined with itself (which is also called Unary relationships), especially when the table has a FOREIGN KEY which references its own PRIMARY KEY. To join a table itself means that each row of the table is combined with itself and with every other row of the table.
- The self join can be viewed as a join of two copies of the same table. The table is not actually copied, but SQL performs the command as though it were.

Theta Join:

- This type of join is commonly used in complex queries. If a join uses comparison operator other than equality operator, it is performing theta join operation.

THETA Join

Example:



- **SELECT * FROM** Faculty, Offering
  WHERE Faculty.FacSSM=Offering.FacSSN;

THETA Join

Example:

Car

CarModel	CarPrice
CarA	20'000
CarB	30'000
CarC	50'000

Boat

BoatModel	BoatPrice
Boat1	10'000
Boat2	40'000
Boat3	60'000

$$Car \bowtie Boat$$

$$CarPrice \geq BoatPrice$$

CarModel	CarPrice	BoatModel	BoatPrice
CarA	20'000	Boat1	10'000
CarB	30'000	Boat1	10'000
CarC	50'000	Boat1	10'000
CarC	50'000	Boat2	40'000

- RESULT=Car $\bowtie_{\{CarPrice>BoatPrice\}}$ Boat;
- Result=R1 $\bowtie_{\{Condition\}}$ R2; Condition: $\{<, >, =, \leq, \geq, \neq\}$;
- EquiJoin when "=".
- **SELECT * FROM** Car, Boat
  WHERE CarPrice>BoatPrice;

Exercise 1 for Equi-Join

Figure 6.15
A database state for the relations *T*1 and *T*2.

TABLE T1

P	Q	R
10	a	5
15	b	8
25	a	6

TABLE T2

A	B	C
10	b	6
25	c	3
10	b	5

SQL query Result

a. $T1 \bowtie_{(T1.P=T2.A)} T2$

b. $T1 \bowtie_{(T1.Q=T2.B)} T2$

Exercise 2

Department	Dno	Dname	DHeadSsn	Location

Student	SID	Sname	Dno	SAge

Faculty	FSsn	Fname	Dno	FAge

Write **Relational Algebra** and **SQL queries** for following questions:

- What are the names of students who are from department 'Computer Science'?
- What are the names of faculties who are younger than a student?
- What are the names of faculties who works in 'Keller Hall'?

Anti-Join:

- It is inverse of join. When you would like to keep all the records in the original table except those records that match the other table.
- In the same way, left anti join will be inverse of left outer join.