UNIT IV

JOINS

JOIN Operation

- A JOIN operation is **used to combine data from two relations so that related information can be presented in a single table.**
- This operation is very important for any relational database with more than a single relation because **it allows us to process relationships among relations**.
- **Join operation is a combination of a Cartesian product followed by a selection process which satisfy certain condition.**
    - **Join is a combination of a Cartesian product followed by a selection process.**
- A join operation pairs two tuples from different relations, if and only if a given join condition is satisfied.
- **The JOIN operation is denoted by ⋈ (**bow tie symbol).
- The general form of a JOIN operation on two relations R(A1,A2, ...,An) and S(B1,B2, ...,Bm) is :

R ⋈ <join condition>S

JOIN Operation

- **The result of the JOIN is a relation Q:**
  - with **n+m attributes** Q(A1,A2, ...,An,B1,B2,... ,Bm) in that order;
  - Q has one tuple for each combination of tuples—one from R and one from S—whenever the combination satisfies the join condition.
- **Difference between CARTESIAN PRODUC T and JOIN**.
  - In JOIN, only combinations of tuples satisfying the join condition appear in the result, whereas in the CARTESIAN PRODUCT all combinations of tuples are included in the result.
- The join condition is specified on attributes from the two relations R and S and is evaluated for each combination of tuples. Each tuple combination for which the join condition evaluates to TRUE is included in the resulting relation Q as a single combined tuple.

JOIN Operation

- If no combination of tuples satisfies the join condition, the result of a JOIN is an empty relation with zero tuples.
- In general, if R has n_R tuples and S has n_s tuples, the result of a JOIN operation R<join condition> S will have **between zero and $n_R * n_s$ tuples**.
- The expected size of the join result divided by the maximum size **$n_R * n_s$** leads to a ratio called **join selectivity**, which is a property of each join condition
- **If there is no join condition**, all combinations of tuples qualify and the JOIN degenerates into a CARTESIAN PRODUCT, also called CROSS PRODUCT or CROSS JOIN.

JOIN Operation - Example

STUDENT

SID	NAME	STD
101	Alex	10
102	Maria	11

SUBJECTS

CLASS	SUBJECT
10	Math
10	English
11	Music
11	Sports

STUDENT

SID	NAME	STD
101	Alex	10
102	Maria	11

SUBJECTS

CLASS	SUBJECT
10	Math
10	English
11	Music
11	Sports

STUDENT X SUBJECTS

SID	NAME	STD	CLASS	SUBJECT
101	Alex	10	10	Math
101	Alex	10	10	English
101	Alex	10	11	Music
101	Alex	10	11	Sports
102	Maria	11	10	Math
102	Maria	11	10	English
102	Maria	11	11	Music
102	Maria	11	11	Sports

JOIN Operation - Example

STUDENT

SID	NAME	STD
101	Alex	10
102	Maria	11

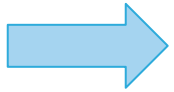
SUBJECTS

CLASS	SUBJECT
10	Math
10	English
11	Music
11	Sports

STUDENT ⋈ Student.Std = Subject.Class **SUBJECT**

STUDENT X SUBJECTS

SID	NAME	STD	CLASS	SUBJECT
101	Alex	10	10	Math
101	Alex	10	10	English
101	Alex	10	11	Music
101	Alex	10	11	Sports
102	Maria	11	10	Math
102	Maria	11	10	English
102	Maria	11	11	Music
102	Maria	11	11	Sports



STUDENT X SUBJECTS

SID	NAME	STD	CLASS	SUBJECT
101	Alex	10	10	Math
101	Alex	10	10	English
102	Maria	11	11	Music
102	Maria	11	11	Sports

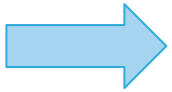
JOIN Operation - Example

EMPLOYEE	
EMP_CODE	EMP_NAME
101	STEPHAN
102	JACK
103	HARRY

SALARY	
EMP_CODE	SALARY
101	50000
102	30000
103	25000

EMPLOYEE $\bowtie$ **EMPLOYEE.EMP_CODE= SALAY.EMP_CODE** SALARY

EMPLOYEE X SALARY			
EMP_CODE	EMP_NAME	EMP_CODE	SALARY
101	STEPHAN	101	50000
101	STEPHAN	102	30000
101	STEPHAN	103	25000
102	JACK	101	50000
102	JACK	102	30000
102	JACK	103	25000
103	HARRY	101	50000
103	HARRY	102	30000
103	HARRY	103	25000



EMP_CODE	EMP_NAME	EMP_CODE	SALARY
101	STEPHAN	101	50000
102	JACK	102	30000
103	HARRY	103	25000

JOIN Operation - TYPES

- A Join operation combines two tuples from two different relations, if and only if a given condition is satisfied..
- There are mainly two types of join which are further divided as follows:

1) Inner Join

- Natural Join
- Theta Join
- Equi Join

2) Outer Join

- Left Outer Join
- Right Inner Join
- Full Outer Join

JOIN Operation – INNER JOINS

1) Inner Join

- **Inner join is a type of join in which only those tuples are selected which fulfill the required conditions.**
- All those tuples which do not satisfy the required conditions are excluded.

❖ **Natural Join (⋈)**

- We can perform a Natural Join only if there is at least one common attribute that exists between two relations. In addition, the attributes must have the same name and domain.
- Natural Join is a join which is performed if there is a common attribute between the relations.
- **Notation:** **R1 ⋈ R2** where R1 and R2 are two relations.
- Natural join acts on those matching attributes where the values of attributes in both the relations are same

JOIN Operation - TYPES

Natural Join (⋈)

Courses		
CID	Course	Dept
CS01	Database	CS
ME01	Mechanics	ME
EE01	Electronics	EE

HoD	
Dept	Head
CS	Alex
ME	Maya
EE	Mira

Courses ⋈ HoD			
Dept	CID	Course	Head
CS	CS01	Database	Alex
ME	ME01	Mechanics	Maya
EE	EE01	Electronics	Mira

Student

S_id	Name	Class	Age	C_id
1	Andrew	5	25	11
2	Angel	10	30	11
3	Anamika	8	35	22

Course

C_id	C_name
11	Foundation C
21	C++

Student ⋈ Course

S_id	Name	Class	Age	C_id	C_name
1	Andrew	5	25	11	Foundation C
2	Angel	10	30	11	Foundation C

JOIN Operation - TYPES

❖ **Theta Join**

- This is same as EQUI JOIN but it allows all other operators like >, <, >= etc.
- Theta join is a join which combines the tuples from different relations according to the given theta condition.
- The join condition in theta join is denoted by theta (θ) symbol.
- **This join uses all kind of comparison operator**.
- **Notation** : $R1 \bowtie_{\theta} R2$ where R1 and R2 are relations such that they **don't have any common attribute**.

JOIN Operation - TYPES

❖ Theta Join

Student		
Stud_ID	**Name**	**Standard**
1	Ram	10
2	Shyam	11

Subjects	
Class	**Subject**
10	Math
10	English
11	Music
11	Sports

$$\text{STUDENT} \bowtie_{\text{Student.Std = Subject.Class}} \text{SUBJECT}$$

SID	Name	Std	Class	Subject
1	Ram	10	10	Math
1	Ram	10	10	English
2	Shyam	11	11	Music
2	Shyam	11	11	Sports

JOIN Operation - TYPES

❖ **Equi Join**

- Equi Join is a type of theta join where we use only the equality operator, unlike theta join where we can use any operator.
- When Theta join uses equality operator for comparison, then it is called equi join. The above example of theta join is applicable for equi join..

JOIN Operation - TYPES

❖ Equi Join

Student

Stud_ID	Name	Standard
1	Ram	10
2	Shyam	11

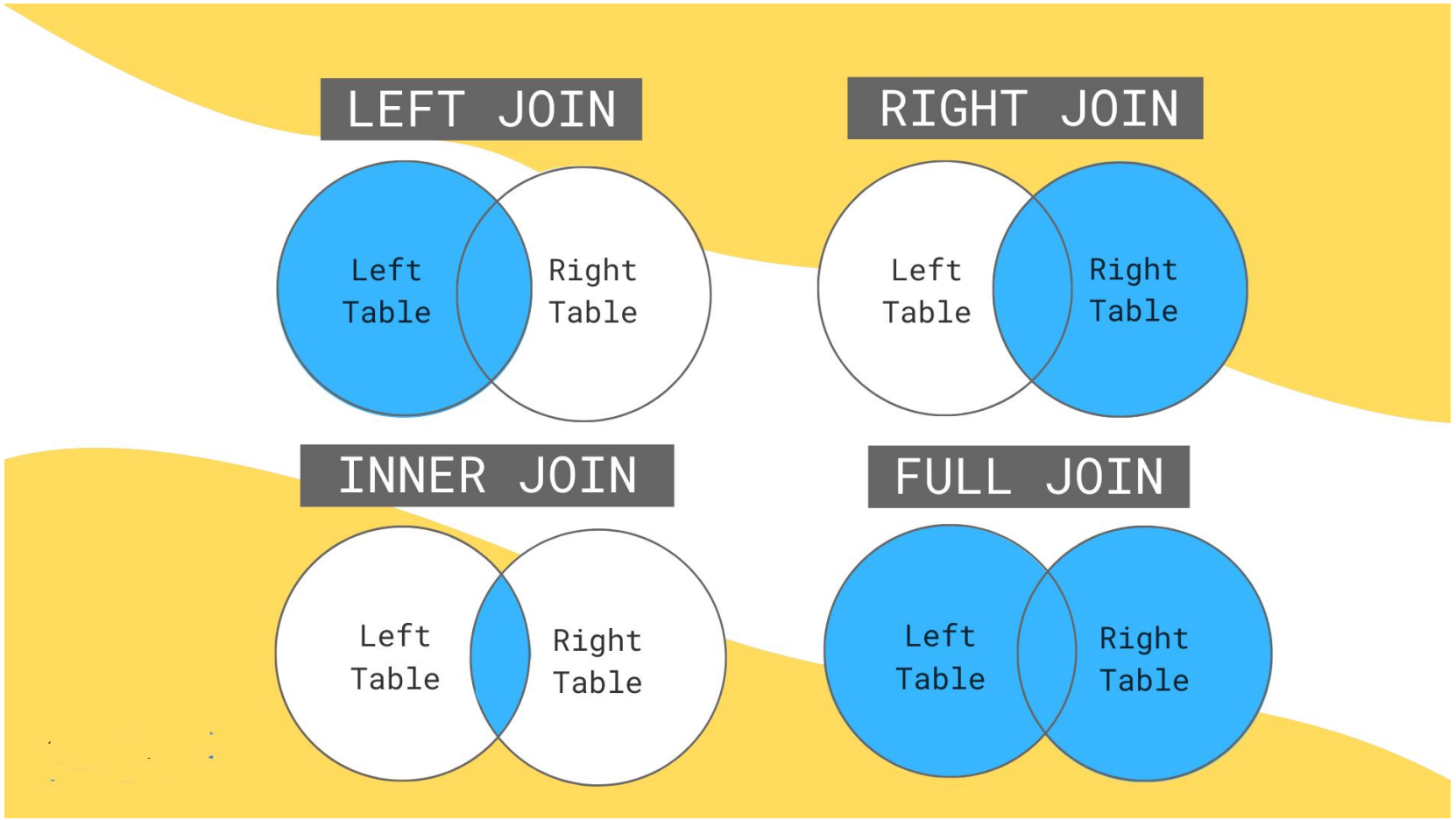
Subjects

Class	Subject
10	Math
10	English
11	Music
11	Sports

$$\text{STUDENT} \bowtie_{\text{Student.Std = Subject.Class}} \text{SUBJECT}$$

SID	Name	Std	Class	Subject
1	Ram	10	10	Math
1	Ram	10	10	English
2	Shyam	11	11	Music
2	Shyam	11	11	Sports

JOIN Operation - TYPES



JOIN Operation – OUTER JOINS

2) Outer Join

- In Inner Join, matched rows are returned and unmatched rows are not returned.
- In an outer join, along with tuples that satisfy the matching criteria, we also include some or all tuples that do not match the criteria.
- Based on the tuples that are added from left, right or both the tables, the outer join is further divided into three types.
    - Left Outer Join
    - Right Inner Join
    - Full Outer Join

CARTESIAN/ CROSS PRODUCT

`shoes`

id	size	color_id
1	seven	2
2	eight	2
3	nine	2
4	seven	1
5	nine	1
6	seven	3
7	ten	NULL

`color`

id	name
1	yellow
2	green
3	pink

id	size	color_id	id	name
1	seven	2	1	yellow
2	eight	2	1	yellow
3	nine	2	1	yellow
4	seven	1	1	yellow
5	nine	1	1	yellow
6	seven	3	1	yellow
7	ten	NULL	1	yellow
1	seven	2	2	green
2	eight	2	2	green
3	nine	2	2	green
4	seven	1	2	green
5	nine	1	2	green
6	seven	3	2	green
7	ten	NULL	2	green
1	seven	2	3	pink
2	eight	2	3	pink
3	nine	2	3	pink
4	seven	1	3	pink
5	nine	1	3	pink
6	seven	3	3	pink
7	ten	NULL	3	pink

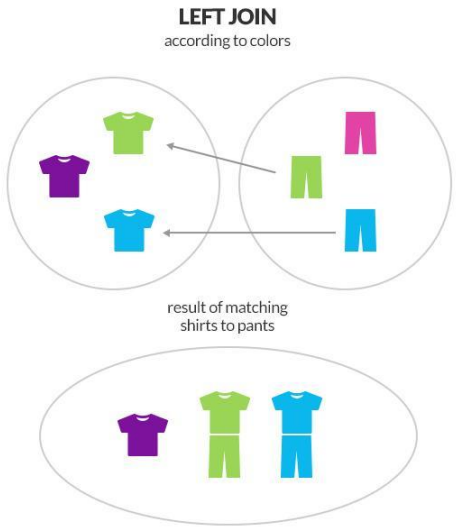
LEFT OUTER JOIN

- The LEFT OUTER JOIN retrieves all records from the first (left) table and matches them to records from the second (right) table. Any non-matching records from the left table are also selected, but with NULL values where the right table records would be.
- ***Notation:*** **R1⟕ R2** where R1 and R2 are relations
- **SHIRT** **PANTS**

SHIRT ⟕ shirt.color_shirt= pants.color_pants **PANTS**

color_shirt
yellow
green
blue

color_pants
pink
green
blue



color_shirt	color_pants
yellow	NULL
green	green
blue	blue

- **With LEFT JOIN, all shirts and any matching pants were returned.**

JOIN Operation – OUTER JOINS

❖ Left Outer Join (⟕)

- Left Outer Join is a type of join in which all the tuples from left relation are included and only those tuples from right relation are included which have a common value in the common attribute on which the join is being performed..

Student

S_id	Name	Class	Age	C_type
1	Andrew	5	25	A
2	Angel	10	30	A
3	Anamika	8	35	C

Course

C_type	C_name
A	Foundation C
B	C++

Student ⟕ Course

S_id	Name	Class	Age	C_type	C_name
1	Andrew	5	25	A	Foundation C
2	Angel	10	30	A	Foundation C
3	Anamika	8	35	C	-

We can see that the new resulting table has all the tuples from the Student table but it doesn't have that tuple from the course table whose attributes values was not matching. Also, it fills the table with the null value for those columns whose value is not defined.

JOIN Operation – OUTER JOINS

❖ Left Outer Join (⟕)

EMPLOYEE

EMP_NAME	STREET	CITY
Ram	Civil line	Mumbai
Varun	S.G.Road	Kolkata
Lakshmi	C.G.Road	Delhi
Hari	AnandNagar	Hyderabad

FACT_WORKERS

EMP_NAME	BRANCH	SALARY
Ram	Infosys	10000
Varun	Wipro	20000
Neha	HCL	30000
Hari	TCS	50000

EMPLOYEE ⟕ FACT_WORKERS

Output:

EMP_NAME	STREET	CITY	BRANCH	SALARY
Ram	Civil line	Mumbai	Infosys	10000
Varun	S.G.Road	Kolkata	Wipro	20000
Hari	AnandNagar	Hyderabad	TCS	50000
Lakshmi	C.G.Road	Delhi	NULL	NULL

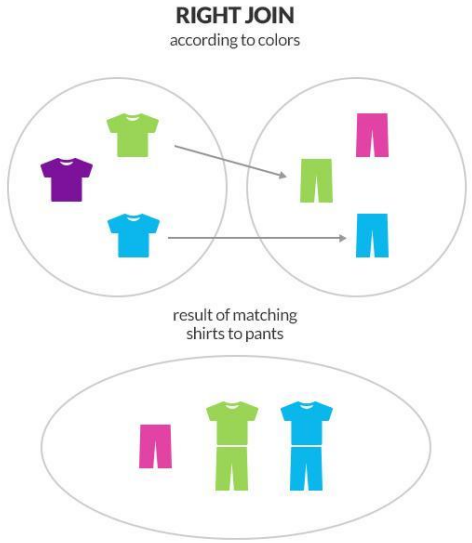
RIGHT OUTER JOIN

- A RIGHT JOIN returns all rows from the right table. If there are no matches in the left table, NULL values are returned for those columns.
- *Notation:* **R1 ⋈ R2** where R1 and R2 are relations
- **SHIRT** **PANTS**

color_shirt
yellow
green
blue

color_pants
pink
green
blue

SHIRT ⋈ shirt.color_shirt= pants.color_pants PANTS



color_shirt	color_pants
NULL	pink
green	green
blue	blue

- **With RIGHT JOIN, all pants and any matching shirts were returned.**

JOIN Operation – OUTER JOINS

❖ Right Outer Join (⟖)

- Right Outer Join is a type of join in which all the tuples from right relation are included and only those tuples from left relation are included which have a common value in the common attribute on which the right join is being performed.

Student

S_id	Name	Class	Age	C_type
1	Andrew	5	25	A
2	Angel	10	30	A
3	Anamika	8	35	C

Course

C_type	C_name
A	Foundation C
B	C++

Student ⟖ Course

S_id	Name	Class	Age	C_type	C_name
1	Andrew	5	25	A	Foundation C
2	Angel	10	30	A	Foundation C
-	-	-	-	B	C++

We can see that the new resulting table has all the tuples from the Course table but it doesn't have that tuple from the Student table whose attributes values was not matching. Also, it fills the table with the null value for those columns whose value is not defined.

JOIN Operation – OUTER JOINS

❖ Right Outer Join (⟖)

EMPLOYEE

EMP_NAME	STREET	CITY
Ram	Civil line	Mumbai
Varun	S.G.Road	Kolkata
Lakshmi	C.G.Road	Delhi
Hari	AnandNagar	Hyderabad

FACT_WORKERS

EMP_NAME	BRANCH	SALARY
Ram	Infosys	10000
Varun	Wipro	20000
Neha	HCL	30000
Hari	TCS	50000

EMPLOYEE ⟖ FACT_WORKERS

Output :

EMP_NAME	BRANCH	SALARY	STREET	CITY
Ram	Infosys	10000	Civil line	Mumbai
Varun	Wipro	20000	S.G.Road	Kolkata
Hari	TCS	50000	AnandNagar	Hyderabad
Neha	HCL	30000	NULL	NULL

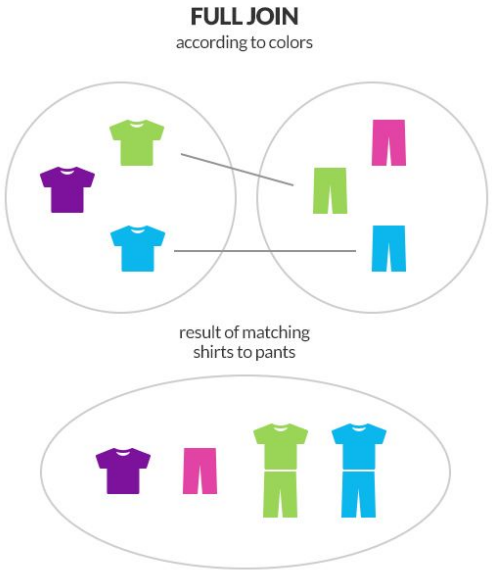
FULL OUTER JOIN

- It shows all records from both tables. If possible, it will match the records; if not, a NULL will be shown where the matching record would be.
- **SHIRT PANTS**

SHIRT ⋈ shirt.color_shirt= pants.color_pants **PANTS**

color_shirt
yellow
green
blue

color_pants
pink
green
blue



color_shirt	color_pants
yellow	NULL
green	green
blue	blue
NULL	pink

- **FULL JOIN returned all possible clothes: all shirts and all pants.**

JOIN Operation – OUTER JOINS

❖ Full Outer Join (⟗)

- Full Outer Join is a type of join in which all the tuples from the left and right relation which are having the same value on the common attribute. Also, they will have all the remaining tuples which are not common on in both the relations.
- ***Notation:*** **R1 ⟗ R2** where R1 and R2 are relations

Student

S_id	Name	Class	Age	C_type
1	Andrew	5	25	A
2	Angel	10	30	A
3	Anamika	8	35	C

Course

C_type	C_name
A	Foundation C
B	C++

Student ⟗ Course

S_id	Name	Class	Age	C_type	C_name
1	Andrew	5	25	A	Foundation C
2	Angel	10	30	A	Foundation C
3	Anamika	8	35	C	-
-	-	-	-	B	C++

We can see that the new resulting table has all the tuples from the Course table as well as the Student table. Also, it fills the table with the null value for those columns whose value is not defined.

JOIN Operation – OUTER JOINS

❖ Full Outer Join (⟗)

EMPLOYEE

EMP_NAME	STREET	CITY
Ram	Civil line	Mumbai
Varun	S.G.Road	Kolkata
Lakshmi	C.G.Road	Delhi
Hari	AnandNagar	Hyderabad

FACT_WORKERS

EMP_NAME	BRANCH	SALARY
Ram	Infosys	10000
Varun	Wipro	20000
Neha	HCL	30000
Hari	TCS	50000

EMPLOYEE ⟗ FACT_WORKERS

EMP_NAME	STREET	CITY	BRANCH	SALARY
Ram	Civil line	Mumbai	Infosys	10000
Shyam	S.G.Road	Kolkata	Wipro	20000
Hari	AnandNagar	Hyderabad	TCS	50000
Lakshmi	C.G.Road	Delhi	NULL	NULL
Neha	NULL	NULL	HCL	30000

OUTER JOIN

- An OUTER JOIN returns all the rows from one table and some or all of the rows from another table. There are three types of outer joins:
- The three kinds of OUTER JOIN:
  - **LEFT OUTER JOIN -** returns every record in the left table and all matching records from the right table. If there's no match found, a NULL is shown next to the unmatched record.
  - **RIGHT OUTER JOIN -** returns every record in the right table and all matching records from the left table. If there's no match found, a NULL is shown next to the unmatched record.
  - **FULL OUTER JOIN -** returns all records from both tables. All unmatched records are paired with NULLs

INNER JOIN

- INNER JOIN combines data from multiple tables by joining them based on a matching record. This kind of join requires a joining condition - it returns only those records that match the join condition in both tables.

shoes ⋈ **shoes.color_id = colors.id** color

`shoes`

id	size	color_id
1	seven	2
2	eight	2
3	nine	2
4	seven	1
5	nine	1
6	seven	3
7	ten	NULL

`color`

id	name
1	yellow
2	green
3	pink

id	size	color_id	id	name
1	seven	**2**	2	green
2	eight	**2**	2	green
3	nine	**2**	2	green
4	seven	**1**	1	yellow
5	nine	**1**	1	yellow
6	seven	**3**	3	pink

Coloured cells are keys, used to match tables

x

		x1
		x2
		x3
		x4

y

		y1
		y2
		y3
		y4

Grey cells are the other variables taken along for the ride

All joins are built on the **cross join** that makes all combinations.

		x1			y1
		x2			y1
		x3			y1
		x4			y1
		x1			y2
		x2			y2
		x3			y2
		x4			y2
		x1			y3
		x2			y3
		x3			y3
		x4			y3
		x1			y4
		x2			y4
		x3			y4
		x4			y4

The **inner join** keeps rows where keys match

		x1			y1
		x2			y2
		x3			y2
		x2			y3
		x3			y3

The duplicated keys are dropped in the final result

		x1	y1
		x2	y2
		x3	y2
		x2	y3
		x3	y3

The **left join** adds rows from x that don't have a match in y

		x1			y1
		x2			y2
		x3			y2
		x2			y3
		x3			y3
		x4			

Values from unmatched rows get filled in with NA

		x1	y1
		x2	y2
		x3	y2
		x2	y3
		x3	y3
		x4	

The **right join** adds rows from y that don't have a match in x

		x1			y1
		x2			y2
		x3			y2
		x2			y3
		x3			y3
					y4

		x1	y1
		x2	y2
		x3	y2
		x2	y3
		x3	y3
			y4

The **full join** adds rows from both x and y

		x1			y1
		x2			y2
		x3			y2
		x2			y3
		x3			y3
					y4
		x4			

		x1	y1
		x2	y2
		x3	y2
		x2	y3
		x3	y3
		x4	
			y4

EXAMPLE

EMP

emp_id	emp_name	dept_id	salary
103	Jack	1	1400
104	John	2	1450
108	Alan	3	1150
107	Ram	NULL	600

DEPT

dept_id	dept_name
1	Sales
2	Finance
3	Accounts
4	Marketing

πemp_id, emp_name, dept_name **(EMP** $\bowtie$ emp.dept_id = dept.dept_id **DEPT)**

emp_id	emp_name	dept_name
103	Jack	Sales
104	John	Finance
108	Alan	Accounts
107	Ram	NULL

LEFT OUTER join includes unmatched rows from the left table

EXAMPLE

EMP

emp_id	emp_name	dept_id	salary
103	Jack	1	1400
104	John	2	1450
108	Alan	3	1150
107	Ram	NULL	600

DEPT

dept_id	dept_name
1	Sales
2	Finance
3	Accounts
4	Marketing

πemp_id, emp_name, dept_name **(EMP** $\bowtie$ emp.dept_id = dept.dept_id **DEPT)**

emp_id	emp_name	dept_name
103	Jack	Sales
104	John	Finance
108	Alan	Accounts
NULL	NULL	Marketing

RIGHT OUTER join includes unmatched rows from the right table