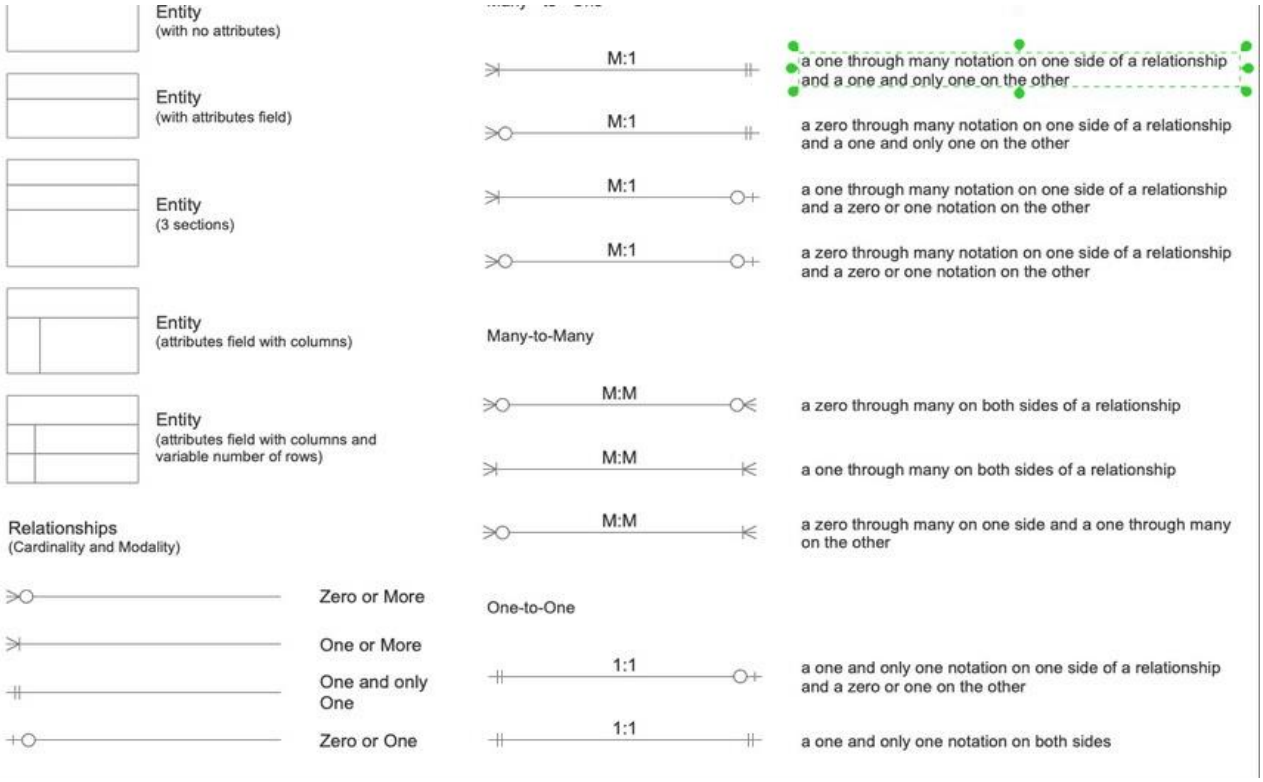
Cardinality	Minimum No. of tables
1:1 cardinality with partial participation of both entities	2
1:1 cardinality with total participation of atleast 1 entity	1
1:n cardinality	2
m:n cardinality	3

This is a general observation. Special cases need to be taken care. We may need extra table if attribute of a relationship can't be moved to any entity side.

Keys of a relation:

There are various types of keys in a relation which are:

- **Candidate Key:**
  The minimal set of attributes which can determine a tuple uniquely. There can be more than 1 candidate key of a relation and its proper subset can't determine tuple uniquely and it can't be NULL.
- **Super Key:**
  The set of attributes which can determine a tuple uniquely. A candidate key is always a super key but vice versa is not true.
- **Primary Key and Alternate Key:**
  Among various candidate keys, one key is taken primary key and others are alternate keys.
- **Foreign Key:**
  Foreign Key is a set of attributes in a table which is used to refer the primary key or alternative key of the same or other table.



Relational Algebra: Procedural language with basic and extended operators.

Basic Operator	Semantic
σ(Selection)	Select rows based on given condition
∏(Projection)	Project some columns
X (Cross Product)	Cross product of relations, returns **m*n** rows where m and n are number of rows in R1 and R2 respectively.
U (Union)	Return those tuples which are either in R1 or in R2. Max no. of rows returned **= m+n** and Min no. of rows returned = **max(m,n)**
–(Minus)	R1-R2 returns those tuples which are in R1 but not in R2. Max no. of rows returned = **m** and Min no. of rows returned = **m-n**
ρ(Rename)	Renaming a relation to other relation.

Extended operator	Semantic
Intersection (∩)	Intersection operator when applied on two relations as R1∩R2 will give a relation with tuples which are in R1 as well as R2.

Conditional Join (⋈c)	Conditional Join is used when you want to join two or more relation based on some conditions.
Equijoin (⋈)	Equijoin is a special case of conditional join where only equality condition holds between a pair of attributes.
Natural Join(⋈)	It is a special case of equijoin in which equality condition hold on all attributes which have same name in relations R and S (relations on which join operation is applied).
Left Outer Join(⟕)	When applying join on two relations R and S, Left Outer Joins gives all tuples of R in the result set. The tuples of R which do not satisfy join condition will have values as NULL for attributes of S.
Right Outer Join(⟖)	When applying join on two relations R and S, Right Outer Joins gives all tuples of S, in the result set. The tuples of S which do not satisfy join condition will have values as NULL for attributes of R.
Full Outer Join(⟗)	Full Outer Joins gives all tuples of S and all tuples of R in the result set. The tuples of S which do not satisfy join condition will have values as NULL for attributes of R and vice versa.
Division Operator (÷)	

Table 1: STUDENT_SPORTS

ROLL_NO	SPORTS
1	Badminton
2	Cricket
2	Badminton
4	Badminton

Table 2: EMPLOYEE

EMP_NO	NAME	ADDRESS	PHONE	AGE
1	RAM	DELHI	9455123451	18
5	NARESH	HISAR	9782918192	22
6	SWETA	RANCHI	9852617621	21
4	SURESH	DELHI	9156768971	18

Table 3: STUDENT

ROLL_NO	NAME	ADDRESS	PHONE	AGE
1	RAM	DELHI	9455123451	18
2	RAMESH	GURGAON	9652431543	18
3	SUJIT	ROHTAK	9156253131	20
4	SURESH	DELHI	9156768971	18

Table 4 :ALL_SPORTS

SPORTS
Badminton
Cricket

Selection operator (σ): Selection operator is used to select tuples from a relation based on some condition. Syntax:

$\sigma_{(Cond)(Relation\ Name)}$

Extract students whose age is greater than 18 from STUDENT relation given in Table 3

$\sigma_{(AGE>18)(STUDENT)}$

RESULT:

ROLL_NO	NAME	ADDRESS	PHONE	AGE
3	SUJIT	ROHTAK	9156253131	20

Projection Operator (∏): Projection operator is used to project particular columns from a relation. Syntax:

$\Pi_{(Column\ 1,Column\ 2\ldots.Column\ n)(Relation\ Name)}$

Extract ROLL_NO and NAME from STUDENT relation given in Table 3

$\Pi_{(ROLL\_NO,NAME)(STUDENT)}$

RESULT:

ROLL_NO	NAME
1	RAM
2	RAMESH
3	SUJIT
4	SURESH

Note: If resultant relation after projection has duplicate rows, it will be removed. For Example: $\prod_{(ADDRESS)}$(STUDENT) will remove one duplicate row with value DELHI and return three rows.

Cross Product(X): Cross product is used to join two relations. For every row of Relation1, each row of Relation2 is concatenated. If Relation1 has m tuples and and Relation2 has n tuples, cross product of Relation1 and Relation2 will have m X n tuples. Syntax:

`Relation1 X Relation2`

To apply Cross Product on STUDENT relation given in Table 1 and STUDENT_SPORTS relation given in Table 2,

`STUDENT X STUDENT_SPORTS`

RESULT:

ROLL_NO	NAME	ADDRESS	PHONE	AGE	ROLL_NO	SPORTS
1	RAM	DELHI	9455123451	18	1	Badminton
1	RAM	DELHI	9455123451	18	2	Cricket
1	RAM	DELHI	9455123451	18	2	Badminton
1	RAM	DELHI	9455123451	18	4	Badminton
2	RAMESH	GURGAON	9652431543	18	1	Badminton
2	RAMESH	GURGAON	9652431543	18	2	Cricket
2	RAMESH	GURGAON	9652431543	18	2	Badminton
2	RAMESH	GURGAON	9652431543	18	4	Badminton
3	SUJIT	ROHTAK	9156253131	20	1	Badminton
3	SUJIT	ROHTAK	9156253131	20	2	Cricket
3	SUJIT	ROHTAK	9156253131	20	2	Badminton
3	SUJIT	ROHTAK	9156253131	20	4	Badminton
4	SURESH	DELHI	9156768971	18	1	Badminton
4	SURESH	DELHI	9156768971	18	2	Cricket
4	SURESH	DELHI	9156768971	18	2	Badminton
4	SURESH	DELHI	9156768971	18	4	Badminton

Union (U): Union on two relations R1 and R2 can only be computed if R1 and R2 are **union compatible** (These two relation should have same

number of attributes and corresponding attributes in two relations have same domain) . Union operator when applied on two relations R1 and R2 will give a relation with tuples which are either in R1 or in R2. The tuples which are in both R1 and R2 will appear only once in result relation.

Syntax:

```
Relation1 U Relation2
```

Find person who are either student or employee, we can use Union operator like:

```
STUDENT U EMPLOYEE
```

RESULT:

ROLL_NO	NAME	ADDRESS	PHONE	AGE
1	RAM	DELHI	9455123451	18
2	RAMESH	GURGAON	9652431543	18
3	SUJIT	ROHTAK	9156253131	20
4	SURESH	DELHI	9156768971	18
5	NARESH	HISAR	9782918192	22
6	SWETA	RANCHI	9852617621	21

Minus (-): Minus on two relations R1 and R2 can only be computed if R1 and R2 are **union compatible**. Minus operator when applied on two relations as R1-R2 will give a relation with tuples which are in R1 but not in R2. Syntax:

```
Relation1 - Relation2
```

Find person who are student but not employee, we can use minus operator like:

```
STUDENT – EMPLOYEE
```

RESULT:

ROLL_NO	NAME	ADDRESS	PHONE	AGE
2	RAMESH	GURGAON	9652431543	18
3	SUJIT	ROHTAK	9156253131	20

Rename(ρ): Rename operator is used to give another name to a relation.

Syntax:

```
ρ(Relation2, Relation1)
```

To rename STUDENT relation to STUDENT1, we can use rename operator like:

```
ρ(STUDENT1, STUDENT)
```

If you want to create a relation STUDENT_NAMES with ROLL_NO and NAME from STUDENT, it can be done using rename operator as:

$$\rho(\text{STUDENT\_NAMES}, \Pi_{(\text{ROLL\_NO, NAME})}(\text{STUDENT}))$$

Extended operators are those operators which can be derived from basic operators.
There are mainly three types of extended operators in Relational Algebra:

- **Join**
- **Intersection**
- **Divide**

Intersection (∩): Intersection on two relations R1 and R2 can only be computed if R1 and R2 are **union compatible** (These two relation should have same number of attributes and corresponding attributes in two relations have same domain). Intersection operator when applied on two relations as R1∩R2 will give a relation with tuples which are in R1 as well as R2.

Syntax:

$$\text{Relation1} \cap \text{Relation2}$$

Example:

Find a person who is student as well as employee-

$$\text{STUDENT} \cap \text{EMPLOYEE}$$

In terms of basic operators (union and minus) :

$$\text{STUDENT} \cap \text{EMPLOYEE} = \text{STUDENT} + \text{EMPLOYEE} - (\text{STUDENT} \cup \text{EMPLOYEE})$$

RESULT:

ROLL_NO	NAME	ADDRESS	PHONE	AGE
1	RAM	DELHI	9455123451	18
4	SURESH	DELHI	9156768971	18

Conditional Join($\bowtie_c$): Conditional Join is used when you want to join two or more relation based on some conditions.

Example: Select students whose ROLL_NO is greater than EMP_NO of employees

$$\text{STUDENT} \bowtie_{c\ \text{STUDENT.ROLL\_NO}>\text{EMPLOYEE.EMP\_NO}} \text{EMPLOYEE}$$

In terms of basic operators (cross product and selection) :

$$\sigma_{(\text{STUDENT.ROLL\_NO}>\text{EMPLOYEE.EMP\_NO})}(\text{STUDENT} \times \text{EMPLOYEE})$$

RESULT:

ROLL_NO	NAME	ADDRESS	PHONE	AGE	EMP_NO	NAME	ADDRESS	PHONE	AGE
2	RAMESH	GURGAON	9652431543	18	1	RAM	DELHI	9455123451	18
3	SUJIT	ROHTAK	9156253131	20	1	RAM	DELHI	9455123451	18
4	SURESH	DELHI	9156768971	18	1	RAM	DELHI	9455123451	18

Equijoin(⋈): Equijoin is a **special case of conditional join** where only equality condition holds between a pair of attributes. As values of two attributes will be equal in result of equijoin, only one attribute will be appeared in result.
Example:

Select students whose ROLL_NO is equal to EMP_NO of employees

$$STUDENT \bowtie_{STUDENT.ROLL\_NO=EMPLOYEE.EMP\_NO} EMPLOYEE$$

In terms of basic operators (cross product, selection and projection) :

$$\Pi_{(STUDENT.ROLL\_NO,\ STUDENT.NAME,\ STUDENT.ADDRESS,\ STUDENT.PHONE,\ STUDENT.AGE\ EMPLOYEE.NAME,\ EMPLOYEE.ADDRESS,\ EMPLOYEE.PHONE,\ EMPLOYEE>AGE)}(\sigma_{(STUDENT.ROLL\_NO=EMPLOYEE.EMP\_NO)}\ (STUDENT \times EMPLOYEE))$$

RESULT:

ROLL_NO	NAME	ADDRESS	PHONE	AGE	NAME	ADDRESS	PHONE	AGE
1	RAM	DELHI	9455123451	18	RAM	DELHI	9455123451	18
4	SURESH	DELHI	9156768971	18	SURESH	DELHI	9156768971	18

Natural Join(⋈): It is a special case of equijoin in which equality condition hold on all attributes which have same name in relations R and S (relations on which join operation is applied). While applying natural join on two relations, there is no need to write equality condition explicitly. Natural Join will also return the similar attributes only once as their value will be same in resulting relation.
Example:

Select students whose ROLL_NO is equal to ROLL_NO of STUDENT_SPORTS as:

$$STUDENT \bowtie STUDENT\_SPORTS$$

In terms of basic operators (cross product, selection and projection) :

$$\Pi_{(STUDENT.ROLL\_NO,\ STUDENT.NAME,\ STUDENT.ADDRESS,\ STUDENT.PHONE,\ STUDENT.AGE\ STUDENT\_SPORTS.SPORTS)}(\sigma_{(STUDENT.ROLL\_NO=STUDENT\_SPORTS.ROLL\_NO)}\ (STUDENT \times STUDENT\_SPORTS))$$

RESULT:

ROLL_NO	NAME	ADDRESS	PHONE	AGE	SPORTS
1	RAM	DELHI	9455123451	18	Badminton
2	RAMESH	GURGAON	9652431543	18	Cricket
2	RAMESH	GURGAON	9652431543	18	Badminton
4	SURESH	DELHI	9156768971	18	Badminton

Natural Join is by default inner join because the tuples which does not satisfy the conditions of join does not appear in result set. e.g.; The tuple having

ROLL_NO 3 in STUDENT does not match with any tuple in STUDENT_SPORTS, so it has not been a part of result set.

Left Outer Join(⟕): When applying join on two relations R and S, some tuples of R or S does not appear in result set which does not satisfy the join conditions. But Left Outer Joins gives all tuples of R in the result set. The tuples of R which do not satisfy join condition will have values as NULL for attributes of S.
Example:Select students whose ROLL_NO is greater than EMP_NO of employees and details of other students as well

$$\text{STUDENT} \text{\&\#10197}_{\text{STUDENT.ROLL\_NO>EMPLOYEE.EMP\_NO}} \text{EMPLOYEE}$$

RESULT

ROLL_NO	NAME	ADDRESS	PHONE	AGE	EMP_NO	NAME	ADDRESS	PHONE	AGE
2	RAMESH	GURGAON	9652431543	18	1	RAM	DELHI	9455123451	18
3	SUJIT	ROHTAK	9156253131	20	1	RAM	DELHI	9455123451	18
4	SURESH	DELHI	9156768971	18	1	RAM	DELHI	9455123451	18
1	RAM	DELHI	9455123451	18	NULL	NULL	NULL	NULL	NULL

Right Outer Join(⟖): When applying join on two relations R and S, some tuples of R or S does not appear in result set which does not satisfy the join conditions. But Right Outer Joins gives all tuples of S in the result set. The tuples of S which do not satisfy join condition will have values as NULL for attributes of R.
Example:

> Select students whose ROLL_NO is greater than EMP_NO of employees and details of other Employees as well

$$\text{STUDENT} ⟖_{\text{STUDENT.ROLL\_NO>EMPLOYEE.EMP\_NO}} \text{EMPLOYEE}$$

RESULT:

ROLL_NO	NAME	ADDRESS	PHONE	AGE	EMP_NO	NAME	ADDRESS	PHONE	AGE
2	RAMESH	GURGAON	9652431543	18	1	RAM	DELHI	9455123451	18
3	SUJIT	ROHTAK	9156253131	20	1	RAM	DELHI	9455123451	18
4	SURESH	DELHI	9156768971	18	1	RAM	DELHI	9455123451	18
NULL	NULL	NULL	NULL	NULL	5	NARESH	HISAR	9782918192	22
NULL	NULL	NULL	NULL	NULL	6	SWETA	RANCHI	9852617621	21
NULL	NULL	NULL	NULL	NULL	4	SURESH	DELHI	9156768971	18

Full Outer Join(⟗): When applying join on two relations R and S, some tuples of R or S does not appear in result set which does not satisfy the join conditions. But Full Outer Joins gives all tuples of S and all tuples of R in the result set. The tuples of S which do not satisfy join condition will have values as NULL for attributes of R and vice versa.
Example:

> Select students whose ROLL_NO is greater than EMP_NO of employees and details of other Employees as well and other Students as well

$$\text{STUDENT} ⟗_{\text{STUDENT.ROLL\_NO>EMPLOYEE.EMP\_NO}} \text{EMPLOYEE}$$

RESULT:

ROLL_NO	NAME	ADDRESS	PHONE	AGE	EMP_NO	NAME	ADDRESS	PHONE	AGE
2	RAMESH	GURGAON	9652431543	18	1	RAM	DELHI	9455123451	18

3	SUJIT	ROHTAK	9156253131	20	1	RAM	DELHI	9455123451	18
4	SURESH	DELHI	9156768971	18	1	RAM	DELHI	9455123451	18
NULL	NULL	NULL	NULL	NULL	5	NARESH	HISAR	9782918192	22
NULL	NULL	NULL	NULL	NULL	6	SWETA	RANCHI	9852617621	21
NULL	NULL	NULL	NULL	NULL	4	SURESH	DELHI	9156768971	18
1	RAM	DELHI	9455123451	18	NULL	NULL	NULL	NULL	NULL

Division Operator (÷): Division operator A÷B can be applied if and only if:

- Attributes of B is proper subset of Attributes of A.
- The relation returned by division operator will have attributes = (All attributes of A – All Attributes of B)
- The relation returned by division operator will return those tuples from relation A which are associated to every B's tuple.

Consider the relation STUDENT_SPORTS and ALL_SPORTS given in Table 2 and Table 3 above.

To apply division operator as

STUDENT_SPORTS÷ ALL_SPORTS

- The operation is valid as attributes in ALL_SPORTS is a proper subset of attributes in STUDENT_SPORTS.
- The attributes in resulting relation will have attributes {ROLL_NO,SPORTS}-{SPORTS}=ROLL_NO
- The tuples in resulting relation will have those ROLL_NO which are associated with all B's tuple {Badminton, Cricket}. ROLL_NO 1 and 4 are associated to Badminton only. ROLL_NO 2 is associated to all tuples of B. So the resulting relation will be:

ROLL_NO
2