MODULE II

RELATIONAL MODEL

STRUCTURE OF RELATIONAL DATABASES

Relational model can represent as a table with columns and rows. Each row is known as a tuple. Each table of the column has a name or attribute.

Domain: It contains a set of atomic values that an attribute can take.

Attribute: It contains the name of a column in a particular table. Each attribute Ai must have a domain, dom(Ai)

Relational instance: In the relational database system, the relational instance is represented by a finite set of tuples. Relation instances do not have duplicate tuples.

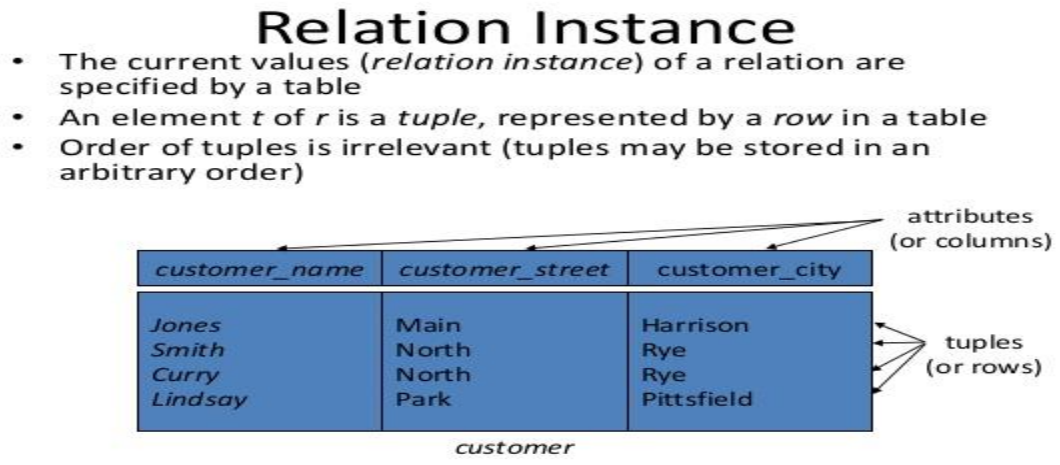
Relational schema: A relational schema contains the name of the relation and name of all columns or attributes.

Relational key: In the relational key, each row has one or more attributes. It can identify the row in the relation uniquely.

Example: STUDENT Relation

NAME	ROLL_NO	PHONE_NO	ADDRESS	AGE
Ram	14795	7305758992	Noida	24
Shyam	12839	9026288936	Delhi	35
Laxman	33289	8583287182	Gurugram	20
Mahesh	27857	7086819134	Ghaziabad	27
Ganesh	17282	9028 9i3988	Delhi	40

- In the given table, NAME, ROLL_NO, PHONE_NO, ADDRESS, and AGE are the attributes.
- The instance of schema STUDENT has 5 tuples.
- t3 = <Laxman, 33289, 8583287182, Gurugram, 20>



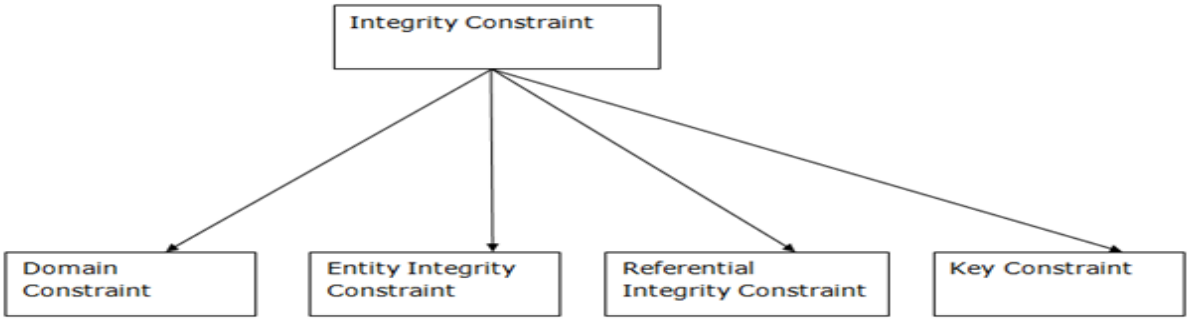
Properties of Relations

- Name of the relation is distinct from all other relations.
- Each relation cell contains exactly one atomic (single) value
- Each attribute contains a distinct name
- Attribute domain has no significance
- tuple has no duplicate value
- Order of tuple can have a different sequence

Integrity Constraints

- Integrity constraints are a set of rules. It is used to maintain the quality of information.
- Integrity constraints ensure that the data insertion, updating, and other processes have to be performed in such a way that data integrity is not affected.
- Thus, integrity constraint is used to guard against accidental damage to the database.

Types of Integrity Constraint



1. Domain constraints

- Domain constraints can be defined as the definition of a valid set of values for an attribute.
- The data type of domain includes string, character, integer, time, date, currency, etc. The value of the attribute must be available in the corresponding domain.

Example:

ID	NAME	SEMENSTER	AGE
1000	Tom	1st	17
1001	Johnson	2nd	24
1002	Leonardo	5th	21
1003	Kate	3rd	19
1004	Morgan	8th	A

Not allowed. Because AGE is an integer attribute

2. Entity integrity constraints

- The entity integrity constraint states that primary key value can't be null.
- This is because the primary key value is used to identify individual rows in relation and if the primary key has a null value, then we can't identify those rows.
- A table can contain a null value other than the primary key field.

Example:

EMPLOYEE

EMP_ID	EMP_NAME	SALARY
123	Jack	30000
142	Harry	60000
164	John	20000
	Jackson	27000

Not allowed as primary key can't contain a NULL value

3. Referential Integrity Constraints

- A referential integrity constraint is specified between two tables.
- In the Referential integrity constraints, if a foreign key in Table 1 refers to the Primary Key of Table 2, then every value of the Foreign Key in Table 1 must be null or be available in Table 2.

Example:

(Table 1)

EMP_NAME	NAME	AGE	D_No
1	Jack	20	11
2	Harry	40	24
3	John	27	18
4	Devil	38	13

Foreign key

Not allowed as D_No 18 is not defined as a Primary key of table 2 and In table 1, D_No is a foreign key defined

Relationships

(Table 2)

Primary Key

D_No	D_Location
11	Mumbai
24	Delhi
13	Noida

4. Key constraints

- Keys are the entity set that is used to identify an entity within its entity set uniquely.
- An entity set can have multiple keys, but out of which one key will be the primary key. A primary key can contain a unique and null value in the relational table.

Example:

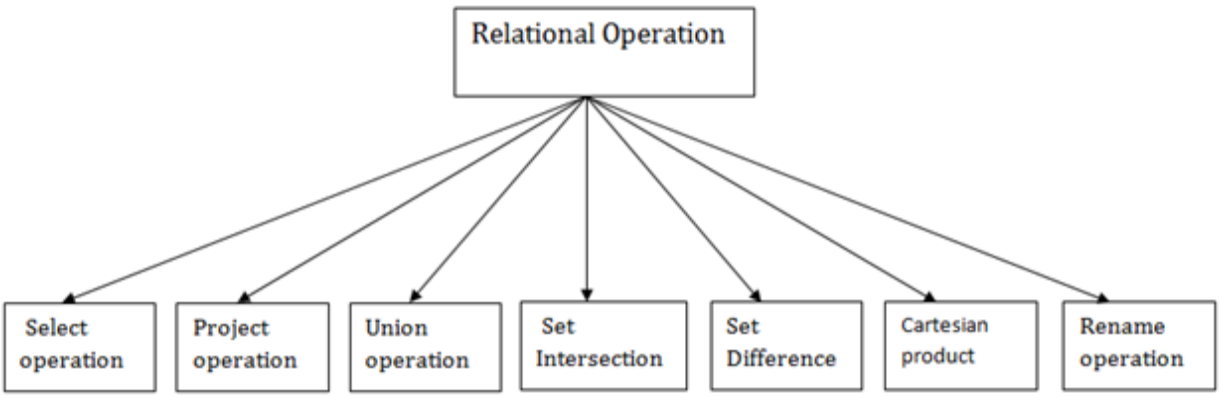
ID	NAME	SEMENSTER	AGE
1000	Tom	1st	17
1001	Johnson	2nd	24
1002	Leonardo	5th	21
1003	Kate	3rd	19
1002	Morgan	8th	22

Not allowed. Because all row must be unique

Relational Algebra

Relational algebra is a procedural query language. It gives a step by step process to obtain the result of the query. It uses operators to perform queries.

Types of Relational operation



1. Select Operation:

- The select operation selects tuples that satisfy a given predicate.
- It is denoted by sigma (σ).

1. Notation: σ p(r)

Where:

σ is used for selection prediction
r is used for relation
p is used as a propositional logic formula which may use connectors like: AND OR and NOT. These relational can use as relational operators like =, ≠, ≥, <, >, ≤.

For example: LOAN Relation

BRANCH_NAME	LOAN_NO	AMOUNT
Downtown	L-17	1000
Redwood	L-23	2000
Perryride	L-15	1500
Downtown	L-14	1500
Mianus	L-13	500
Roundhill	L-11	900
Perryride	L-16	1300

Input:

σ BRANCH_NAME="perryride" (LOAN)

Output:

BRANCH_NAME	LOAN_NO	AMOUNT
Perryride	L-15	1500
Perryride	L-16	1300

2. Project Operation:

- This operation shows the list of those attributes that we wish to appear in the result. Rest of the attributes are eliminated from the table.
- It is denoted by ∏.

Notation: ∏ A1, A2, An (r)

Where

A1, **A2**, **A3** is used as an attribute name of relation **r**.

Example: CUSTOMER RELATION

NAME	STREET	CITY
Jones	Main	Harrison

Smith	North	Rye
Hays	Main	Harrison
Curry	North	Rye
Johnson	Alma	Brooklyn
Brooks	Senator	Brooklyn

Input:

∏ NAME, CITY (CUSTOMER)

Output:

NAME	CITY
Jones	Harrison
Smith	Rye
Hays	Harrison
Curry	Rye
Johnson	Brooklyn
Brooks	Brooklyn

3. Union Operation:

- Suppose there are two tuples R and S. The union operation contains all the tuples that are either in R or S or both in R & S.
- It eliminates the duplicate tuples. It is denoted by ∪.

Notation: R ∪ S

A union operation must hold the following condition:

- R and S must have the attribute of the same number.
- Duplicate tuples are eliminated automatically.

Example:

DEPOSITOR RELATION

CUSTOMER_NAME	ACCOUNT_NO
Johnson	A-101
Smith	A-121
Mayes	A-321
Turner	A-176
Johnson	A-273
Jones	A-472
Lindsay	A-284

BORROW RELATION

CUSTOMER_NAME	LOAN_NO
Jones	L-17
Smith	L-23
Hayes	L-15
Jackson	L-14
Curry	L-93

Smith	L-11
Williams	L-17

Input:

∏ CUSTOMER_NAME (BORROW) ∪ ∏ CUSTOMER_NAME (DEPOSITOR)

Output:

CUSTOMER_NAME
Johnson
Smith
Hayes
Turner
Jones
Lindsay
Jackson
Curry
Williams
Mayes

4. Set Intersection:

- Suppose there are two tuples R and S. The set intersection operation contains all tuples that are in both R & S.
- It is denoted by intersection ∩.

Notation: R ∩ S

Example: Using the above DEPOSITOR table and BORROW table

Input:

∏ CUSTOMER_NAME (BORROW) ∩ ∏ CUSTOMER_NAME (DEPOSITOR)

Output:

CUSTOMER_NAME
Smith
Jones

5. Set Difference:

- Suppose there are two tuples R and S. The set intersection operation contains all tuples that are in R but not in S.
- It is denoted by intersection minus (-).

Notation: R - S

Example: Using the above DEPOSITOR table and BORROW table

Input:

∏ CUSTOMER_NAME (BORROW) - ∏ CUSTOMER_NAME (DEPOSITOR)

Output:

CUSTOMER_NAME
Jackson
Hayes
Willians
Curry

6. Cartesian product

- The Cartesian product is used to combine each row in one table with each row in the other table. It is also known as a cross product.
- It is denoted by X.

Notation: E X D

Example:

EMPLOYEE

EMP_ID	EMP_NAME	EMP_DEPT
1	Smith	A
2	Harry	C
3	John	B

DEPARTMENT

DEPT_NO	DEPT_NAME
A	Marketing
B	Sales
C	Legal

Input:

EMPLOYEE X DEPARTMENT

Output:

EMP_ID	EMP_NAME	EMP_DEPT	DEPT_NO	DEPT_NAME
1	Smith	A	A	Marketing

1	Smith	A	B	Sales
1	Smith	A	C	Legal
2	Harry	C	A	Marketing
2	Harry	C	B	Sales
2	Harry	C	C	Legal
3	John	B	A	Marketing
3	John	B	B	Sales
3	John	B	C	Legal

7. Rename Operation:

The rename operation is used to rename the output relation. It is denoted by **rho** (ρ).

Example: We can use the rename operator to rename STUDENT relation to STUDENT1.

ρ(STUDENT1, STUDENT)

Join Operations:

A Join operation combines related tuples from different relations, if and only if a given join condition is satisfied. It is denoted by ⋈.

Example:

EMPLOYEE

EMP_CODE	EMP_NAME
101	Stephan
102	Jack

103	Harry

SALARY

EMP_CODE	SALARY
101	50000
102	30000
103	25000

Operation: (EMPLOYEE ⋈ SALARY)

Result:

EMP_CODE	EMP_NAME	SALARY
101	Stephan	50000
102	Jack	30000
103	Harry	25000

Types of JOIN:

Various forms of join operation are:

Inner Joins:

- Theta join
- EQUI join
- Natural join

Outer join:

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

Inner Join:

In an inner join, only those tuples that satisfy the matching criteria are included, while the rest are excluded. Let's study various types of Inner Joins:

Theta Join:

The general case of JOIN operation is called a Theta join. It is denoted by symbol **θ**

Example

$A \bowtie_{\theta} B$

Theta join can use any conditions in the selection criteria.

For example:

A ⋈ A.column 2 > B.column 2 (B)	
column 1	**column 2**
1	2

EQUI join:

When a theta join uses only equivalence condition, it becomes a equi join.

It is also known as an inner join. It is the most common join. It is based on matched data as per the equality condition. The equi join uses the comparison operator(=).

Example:

CUSTOMER RELATION

CLASS_ID	NAME
1	John
2	Harry
3	Jackson

PRODUCT

PRODUCT_ID	CITY
1	Delhi
2	Mumbai
3	Noida

Input:

CUSTOMER ⋈ PRODUCT

Output:

CLASS_ID	NAME	PRODUCT_ID	CITY
1	John	1	Delhi
2	Harry	2	Mumbai
3	Harry	3	Noida

EQUI join is the most difficult operations to implement efficiently in an RDBMS and one reason why RDBMS have essential performance problems.

NATURAL JOIN (⋈)

Natural join can only be performed if there is a common attribute (column) between the relations. The name and type of the attribute must be same.

- A natural join is the set of tuples of all combinations in R and S that are equal on their common attribute names.
- It is denoted by ⋈.

Example: Let's use the above EMPLOYEE table and SALARY table:

Input:

∏EMP_NAME, SALARY (EMPLOYEE ⋈ SALARY)

Output:

EMP_NAME	SALARY

Stephan	50000
Jack	30000
Harry	25000

2. Outer Join:

The outer join operation is an extension of the join operation. It is used to deal with missing information.

- An extension of the join operation that avoids loss of information.
- Computes the join and then adds tuples form one relation that does not match tuples in the other relation to the result of the join.

  Uses *null* values:

- *null* signifies that the value is unknown or does not exist
- All comparisons involving *null* are (roughly speaking) **false** by definition.

Example:

EMPLOYEE

EMP_NAME	STREET	CITY
Ram	Civil line	Mumbai
Shyam	Park street	Kolkata
Ravi	M.G. Street	Delhi
Hari	Nehru nagar	Hyderabad

FACT_WORKERS

EMP_NAME	BRANCH	SALARY
Ram	Infosys	10000
Shyam	Wipro	20000
Kuber	HCL	30000
Hari	TCS	50000

Input:

(EMPLOYEE ⋈ FACT_WORKERS)

Output:

EMP_NAME	STREET	CITY	BRANCH	SALARY
Ram	Civil line	Mumbai	Infosys	10000
Shyam	Park street	Kolkata	Wipro	20000
Hari	Nehru nagar	Hyderabad	TCS	50000

An outer join is basically of three types:

a. Left outer join
b. Right outer join
c. Full outer join

a. Left outer join:

- Left outer join contains the set of tuples of all combinations in R and S that are equal on their common attribute names.
- In the left outer join, tuples in R have no matching tuples in S.
- It is denoted by ⟕.

Example: Using the above EMPLOYEE table and FACT_WORKERS table

Input:

1. EMPLOYEE ⋈ FACT_WORKERS

EMP_NAME	STREET	CITY	BRANCH	SALARY
Ram	Civil line	Mumbai	Infosys	10000
Shyam	Park street	Kolkata	Wipro	20000
Hari	Nehru street	Hyderabad	TCS	50000
Ravi	M.G. Street	Delhi	NULL	NULL

b. Right outer join:

- Right outer join contains the set of tuples of all combinations in R and S that are equal on their common attribute names.
- In right outer join, tuples in S have no matching tuples in R.
- It is denoted by ⟖.

Example: Using the above EMPLOYEE table and FACT_WORKERS Relation

Input:

EMPLOYEE ⟖ FACT_WORKERS

Output:

EMP_NAME	BRANCH	SALARY	STREET	CITY
Ram	Infosys	10000	Civil line	Mumbai
Shyam	Wipro	20000	Park street	Kolkata
Hari	TCS	50000	Nehru street	Hyderabad
Kuber	HCL	30000	NULL	NULL

c. Full outer join:

- Full outer join is like a left or right join except that it contains all rows from both tables.

- In full outer join, tuples in R that have no matching tuples in S and tuples in S that have no matching tuples in R in their common attribute name.
- It is denoted by ⟗.

Example: Using the above EMPLOYEE table and FACT_WORKERS table

Input:

EMPLOYEE ⟗ FACT_WORKERS

Output:

EMP_NAME	STREET	CITY	BRANCH	SALARY
Ram	Civil line	Mumbai	Infosys	10000
Shyam	Park street	Kolkata	Wipro	20000
Hari	Nehru street	Hyderabad	TCS	50000
Ravi	M.G. Street	Delhi	NULL	NULL
Kuber	NULL	NULL	HCL	30000

Table 6.1
Operations of Relational Algebra

Operation	Purpose	Notation
SELECT	Selects all tuples that satisfy the selection condition from a relation R.	$\sigma_{\text{<selection condition>}}(R)$
PROJECT	Produces a new relation with only some of the attributes of R, and removes duplicate tuples.	$\pi_{\text{<attribute list>}}(R)$
THETA JOIN	Produces all combinations of tuples from R_1 and R_2 that satisfy the join condition.	$R_1 \bowtie_{\text{<join condition>}} R_2$
EQUIJOIN	Produces all the combinations of tuples from R_1 and R_2 that satisfy a join condition with only equality comparisons.	$R_1 \bowtie_{\text{<join condition>}} R_2$, OR $R_1 \bowtie_{(\text{<join attributes 1>}), (\text{<join attributes 2>})} R_2$
NATURAL JOIN	Same as EQUIJOIN except that the join attributes of R_2 are not included in the resulting relation; if the join attributes have the same names, they do not have to be specified at all.	$R_1 *_{\text{<join condition>}} R_2$, OR $R_1 *_{(\text{<join attributes 1>}), (\text{<join attributes 2>})} R_2$ OR $R_1 * R_2$
UNION	Produces a relation that includes all the tuples in R_1 or R_2 or both R_1 and R_2; R_1 and R_2 must be union compatible.	$R_1 \cup R_2$
INTERSECTION	Produces a relation that includes all the tuples in both R_1 and R_2; R_1 and R_2 must be union compatible.	$R_1 \cap R_2$
DIFFERENCE	Produces a relation that includes all the tuples in R_1 that are not in R_2; R_1 and R_2 must be union compatible.	$R_1 - R_2$
CARTESIAN PRODUCT	Produces a relation that has the attributes of R_1 and R_2 and includes as tuples all possible combinations of tuples from R_1 and R_2.	$R_1 \times R_2$
DIVISION	Produces a relation $R(X)$ that includes all tuples $t[X]$ in $R_1(Z)$ that appear in R_1 in combination with every tuple from $R_2(Y)$, where $Z = X \cup Y$.	$R_1(Z) \div R_2(Y)$

SYNTHESIZING ER DIAGRAM TO RELATIONAL SCHEMA

ER-to-Relational Mapping Algorithm

We will use the COMPANY database example to illustrate the mapping procedure. The COMPANY ER schema is shown in Figure 7.1, and the corresponding COMPANY relational database schema is shown in Figure 7.2 to illustrate the mapping steps

FIGURE 7.1 *from elmasri navathe*

The ER conceptual schema diagram for the COMPANY database.

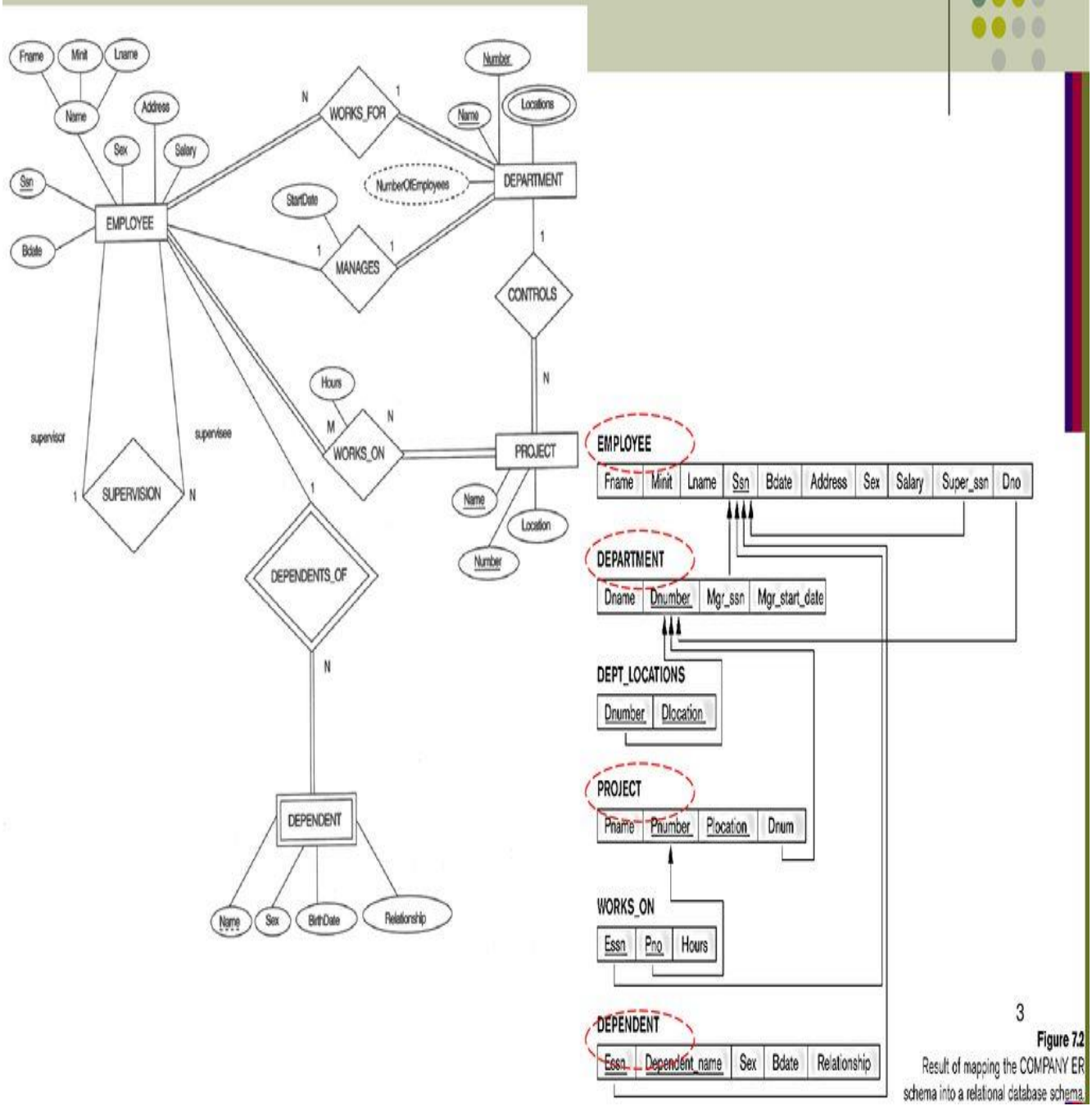


FIGURE 7.1 The ER conceptual schema diagram for the COMPANY databas

Step 1: Mapping of Regular Entity Types.

Step 2: Mapping of Weak Entity Types

Step 3: Mapping of Binary 1:1 Relationship Types

Step 4: Mapping of Binary 1 :N Relationship Types.

Step 5: Mapping of Binary M:N Relationship Types

Step 6: Mapping of Multivalued Attributes

Step 7: Mapping of N-ary Relationship Types.

Steps in detail

Step 1: Mapping of Regular Entity Types.

For each regular (strong) entity type E in the ER schema, create a relation R that **includes all the simple attributes of E. Include only the simple component attributes of a composite attribute**. Choose one of the **key attributes of E as primary key for R**. If the chosen key of E is composite, the set of simple attributes that form it will together form the primary key of R.

In our example, we create the relations EMPLOYEE, DEPARTMENT, and PROJECT in Figure 7.2 tocorrespond to the regular entity types EMPLOYEE, DEPARTMENT, and PROJ ECT from Figure 7.1. The foreign key and relationship attributes, if any, are not included yet; they will be added during subsequent steps. These include the attributes SUPERSSN and DNO of EMPLOYEE, MGRSSN and MGRSTARTDATE of DEPARTMENT, and DNUM of PROJECT.

In our example, we choose SSN, DNUMBER, and PNUMBER as primary keys for the relations EMPLOYEE, DEPARTMENT, and PROJECT, respectively.

Step 2: Mapping of Weak Entity Types

For each weak entity type W in the ER schema with owner entity type E, create a relation R and **include all simple attributes or simple components of composite attribute of W** as attributes of R. In addition**, include as foreign key attributes of R the primary key attribute of the relation that correspond to the owner entity tvpe**: this takes care of the identifying relationship type of W **The primary key** of R is the **combination** of the **primary key of the owner** and **the partial key of the weak entity type W.**

In our example, we create the relation DEPENDENT in this step to correspond to the weak entity type DEPENDENT. We include the primary key SSN of the EMPLOYEE relation-which corresponds to the owner entity type-as a foreign key attribute of DEPENDENT; we renamed it ESSN, although this is not necessary. The primary key of the DEPENDENT relation is the combination {ESSN, DEPENDENT_NAME} because DEPENDENT_NAME is the partial key of DEPENDENT.

Step 3: Mapping of Binary 1:1 Relationship Types.

For each binary 1:1 relationship type R in the ER schema, identify the relations S and T that correspond to the entity types participating in R.

Choose one of the relations-S, say-and include as a **foreign key in S the primary key of T.** It is better to choose an entity type with total participation in R in the role of S. Include all the simple attributes (or simple components of composite attributes) of the 1:1 relationship type R as attributes of S.

In our example, we map the 1:1 relationship type MANAGES from Figure 7.1 by choosing the participating entity type DEPARTMENT to serve in the role of S, because its participation in the MANAGES relationship type is total (every department has a manager). We include the primary key of the EMPLOYEE relation as foreign key in the DEPARTMENT relation and rename it MGRSSN. We also include the simple attribute STARTDATE of the MANAGES relationship type in the DEPARTMENT relation and rename it MGRSTARTDATE.

Step 4: Mapping of Binary 1 :N Relationship Types.

For each regular binary 1:N relationship type R, identify the relation S that represents the participating entity type at the N-side of the relationship type. Include as foreign key in S the primary key of the relation T that represents the other entity type participating in R; this is done because each entity instance on the N-side is related to at most one entity instance on the 1-side of the relationship type. Include any simple attributes (or simple components of composite attributes) of the 1:N relationship type as attributes of S.

In our example, we now map the I:N relationship types WORKS_FOR, CONTROLS, and SUPERVISION from Figure 7.1. For WORKS_FOR we include the primary key DNUMBER of the DEPARTMENT relation as foreign key in the EMPLOYEE relation and call it DNO. For

SUPERVISION we include the primary key of the EMPLOYEE relation as foreign key in the EMPLOYEE relation itself because the relationship is recursive-and call it SUPERSSN. The CONTROLS relationship is mapped to the foreign key attribute DNUM of PROJECT, which references the primary key DNUMBER of the DEPARTMENT relation.

Step 5: Mapping of Binary M:N Relationship Types.

For each binary M:N relationship type R, create a new relation S to represent R. Include as foreign key attributes in S the primary keys of the relations that represent the participating entity types; their combination will form the primary key of S. Also include any simple attributes of the M:N relationship type (or simple components of composite attributes) as attributes of S. Notice that we cannot represent an M:N relationship type by a single foreign key attribute in one ofthe participating relations (as we did for 1:1 or I:N relationship types) because of the M:N cardinality ratio; we must create a separate relationship relation S.

In our example, we map the M:N relationship type WORKS_ON from Figure 7.1 by creating the relation WORKS_ON in Figure 7.2. We include the primary keys of the PROJECT and EMPLOYEE relations as foreign keys in WORKS_ON and rename them PNO and ESSN, respectively. We also include an attribute HOURS in WORKS_ON to represent the HOURS attribute of the relationship type. The primary key of the WORKS_ON relation is the combination of the foreign key attributes {ESSN, PNO}.

Step 6: Mapping of Multivalued Attributes.

For each multivalued attribute A, create a new relation R. This relation R will include an attribute corresponding to A, plus the primary key attribute K-as a foreign key in R-of the relation that represents the entity type or relationship type that has A as an attribute. The primary key of R is the combination of A and K. If the multivalued attribute is composite, we include its simple components.

In our example, we create a relation DEPT_LOCATIONS. The attribute DLOCATION represents the multivalued attribute LOCATIONS of DEPARTMENT, while DNUMBER-as foreign keyrepresents the primary key of the DEPARTMENT relation. The primary key of DEPT_LOCATIONS is the combination of {DNUMBER, DLOCATION}. A separate tuple will exist in DEPT_LOCATIONS for each location that a department has.

Step 7: Mapping of N-ary Relationship Types.

For each n-ary relationship type R, where n > 2, create a new relation S to represent R. Include as foreign key attributes in S the primary keys of the relations rhat represent rhe participating entity types. Also include any simple attributes of the n-ary relationship type (or simple components ofcomposite attributes) as attributes of S. The primary key of S is usually a combinationof all the foreign keys that reference the relations representing the participating entity types.

For example, consider the relationship type SUPPLY of Figure 4.11a. This can be mapped to the relation SUPPLY shown in Figure 7.3, whose primary key is the combination ofthe three foreign keys {SNAME, PARTNO, PROJNAME}.

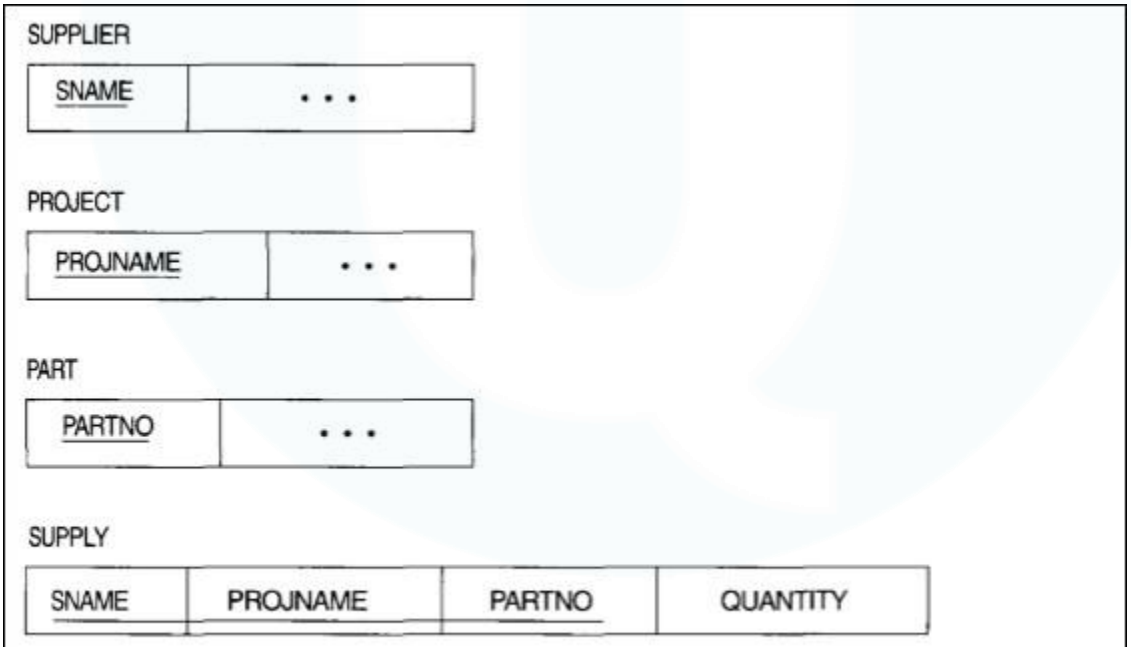


FIGURE 7.3 Mapping the *n*-ary relationship type SUPPLY from Figure 4.11a.

ER diagram-

- ER diagram or **Entity Relationship diagram** is a conceptual model that gives the graphical representation of the logical structure of the database.
- It shows all the constraints and relationships that exist among the different components.

Components of ER diagram-

An ER diagram is mainly composed of following three components-

1. Entity Sets
2. Attributes
3. Relationships sets

Example-

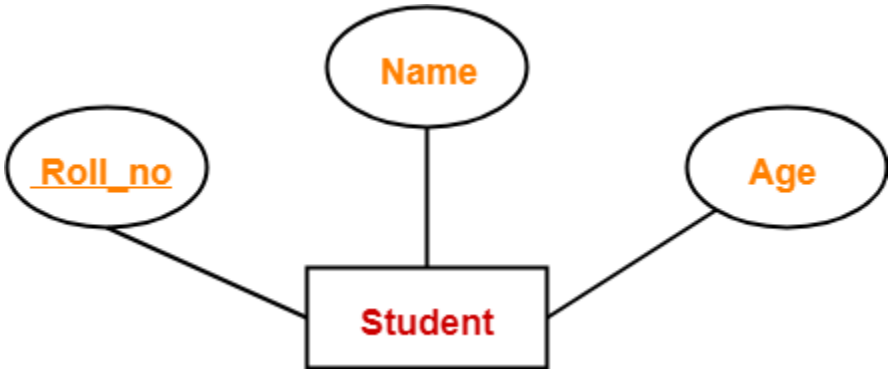
Consider the following Student table-

Roll_no	**Name**	**Age**
1	Akshay	20
2	Rahul	19
3	Pooja	20
4	Aarti	19

This complete table is referred to as "Student Entity Set" and each row represents an "entity".

Representation as ER Diagram-

The above table may be represented as ER diagram as-



Here,

- Roll_no is a primary key that can identify each entity uniquely.
- Thus, by using student's roll number, a student can be identified uniquely.

ER Diagram Symbols-

An ER diagram is composed of several components and each component in ER diagram is represented using a specific symbol.

ER diagram symbols are discussed below-

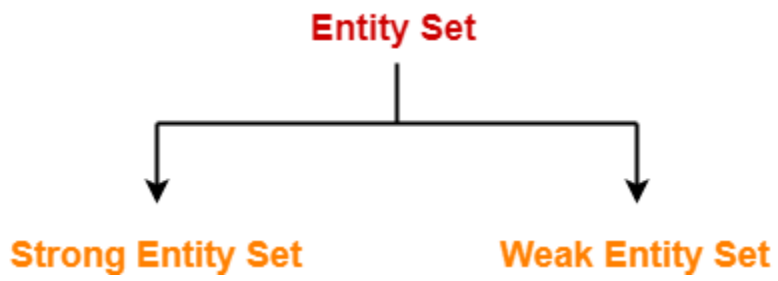
1. For Entity Sets-

An entity set is a set of same type of entities.

An entity refers to any object having-

- Either a physical existence such as a particular person, office, house or car.
- Or a conceptual existence such as a school or a company.

An entity set may be of the following two types-



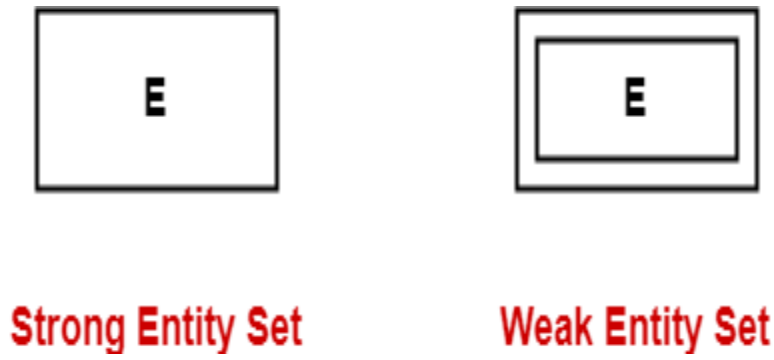
1. Strong entity set
2. Weak entity set

1. Strong Entity Set-

- A strong entity set possess its own primary key.
- It is represented using a single rectangle.

2. Weak Entity Set-

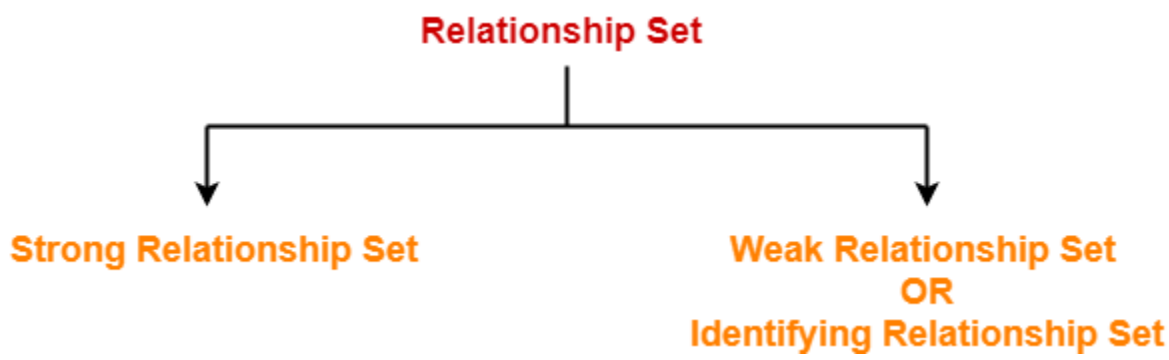
- A weak entity set do not possess its own primary key.
- It is represented using a double rectangle.



2. For Relationship Sets-

- Relationship defines an association among several entities.
- A relationship set is a set of same type of relationships.

A relationship set may be of the following two types-



1. Strong relationship set
2. Weak relationship set

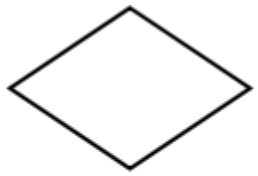
1. Strong Relationship Set-

- A strong relationship exists between two strong entity sets.
- It is represented using a diamond symbol.

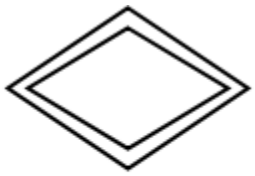
2. Weak Relationship Set-

- A weak or identifying relationship exists between the strong and weak entity set.

- It is represented using a double diamond symbol.



Strong Relationship Set

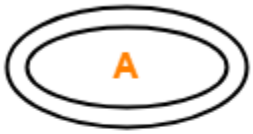


Weak or Identifying Relationship Set

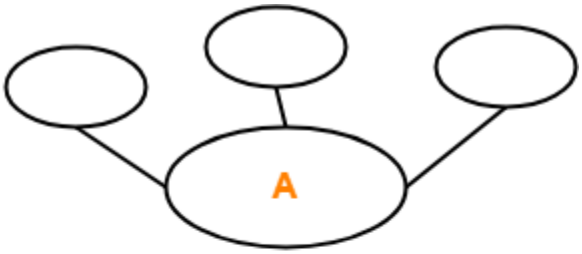
3. For Attributes-

- Attributes are the properties which describes the entities of an entity set.
- There are several types of attributes.

Attribute



Multivalued Attribute



Composite Attribute

Key Attribute

Partial Attribute

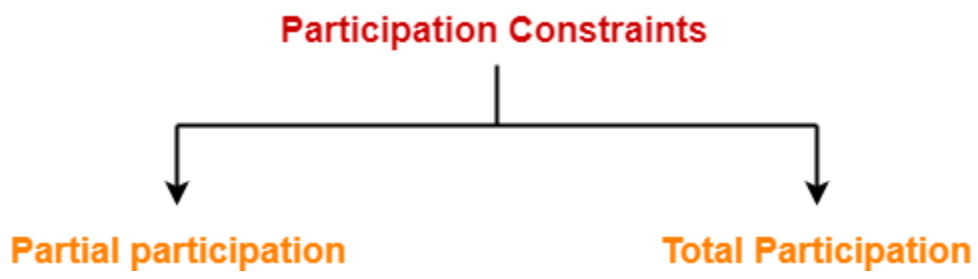
Derived Attribute

Read more- Attributes in ER Diagram

4. For Participation Constraints-

Participation constraint defines the least number of relationship instances in which an entity has to necessarily participate.

There are two types of participation constraints-



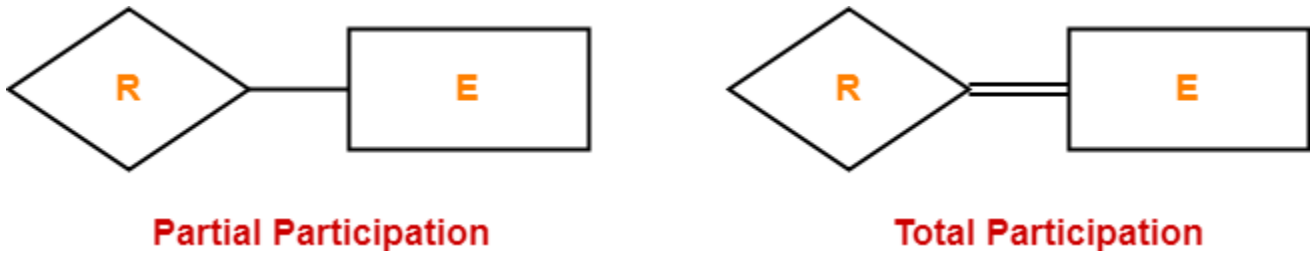
1. Partial participation
2. Total participation

1. Partial Participation-

Partial participation is represented using a single line between the entity set and relationship set.

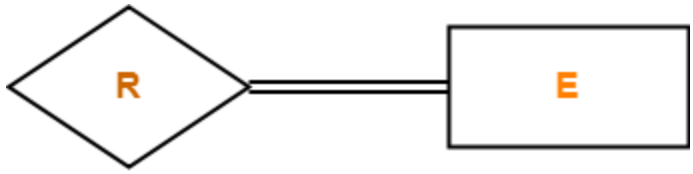
2. Total Participation-

Total participation is represented using a double line between the entity set and relationship set.



1. Total Participation-

- It specifies that each entity in the entity set must compulsorily participate in at least one relationship instance in that relationship set.
- That is why, it is also called as **mandatory participation.**
- Total participation is represented using a double line between the entity set and relationship set.



Total Participation

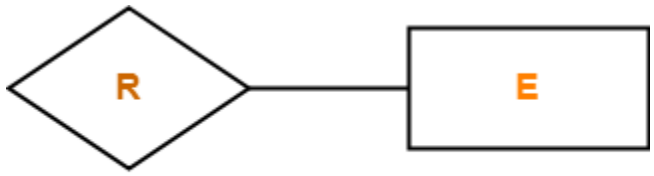
Example-

Here,

- Double line between the entity set "Student" and relationship set "Enrolled in" signifies total participation.
- It specifies that each student must be enrolled in at least one course.

2. Partial Participation-

- It specifies that each entity in the entity set may or may not participate in the relationship instance in that relationship set.
- That is why, it is also called as **optional participation.**
- Partial participation is represented using a single line between the entity set and relationship set.



Partial Participation

Example-

Here,

- Single line between the entity set "Course" and relationship set "Enrolled in" signifies partial participation.
- It specifies that there might exist some courses for which no enrollments are made.

Relationship between Cardinality and Participation Constraints-

Minimum cardinality tells whether the participation is partial or total.

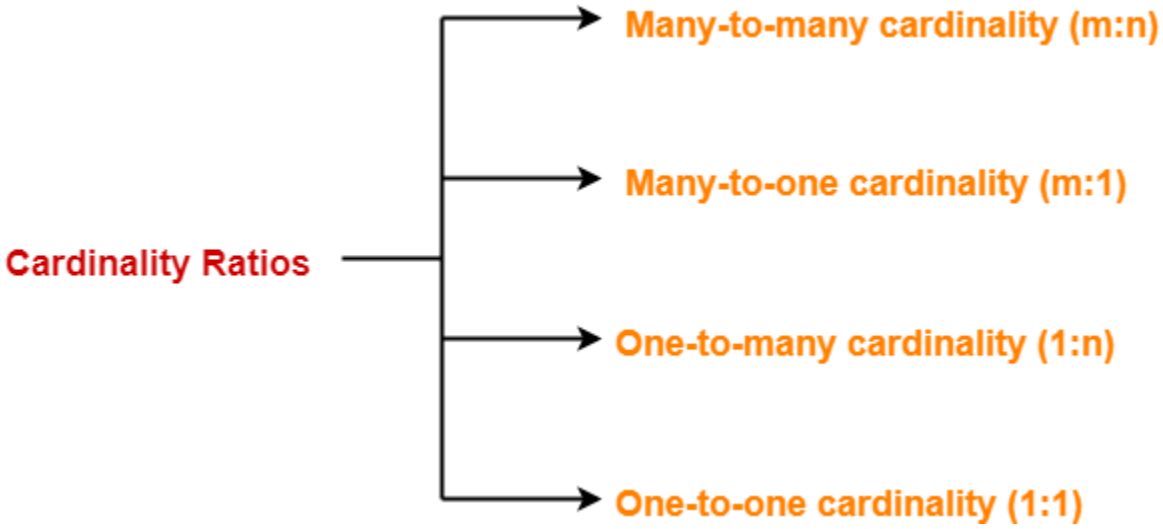
- If minimum cardinality = 0, then it signifies partial participation.
- If minimum cardinality = 1, then it signifies total participation.

Maximum cardinality tells the maximum number of entities that participates in a relationship set.

6. For Cardinality Constraints / Ratios-

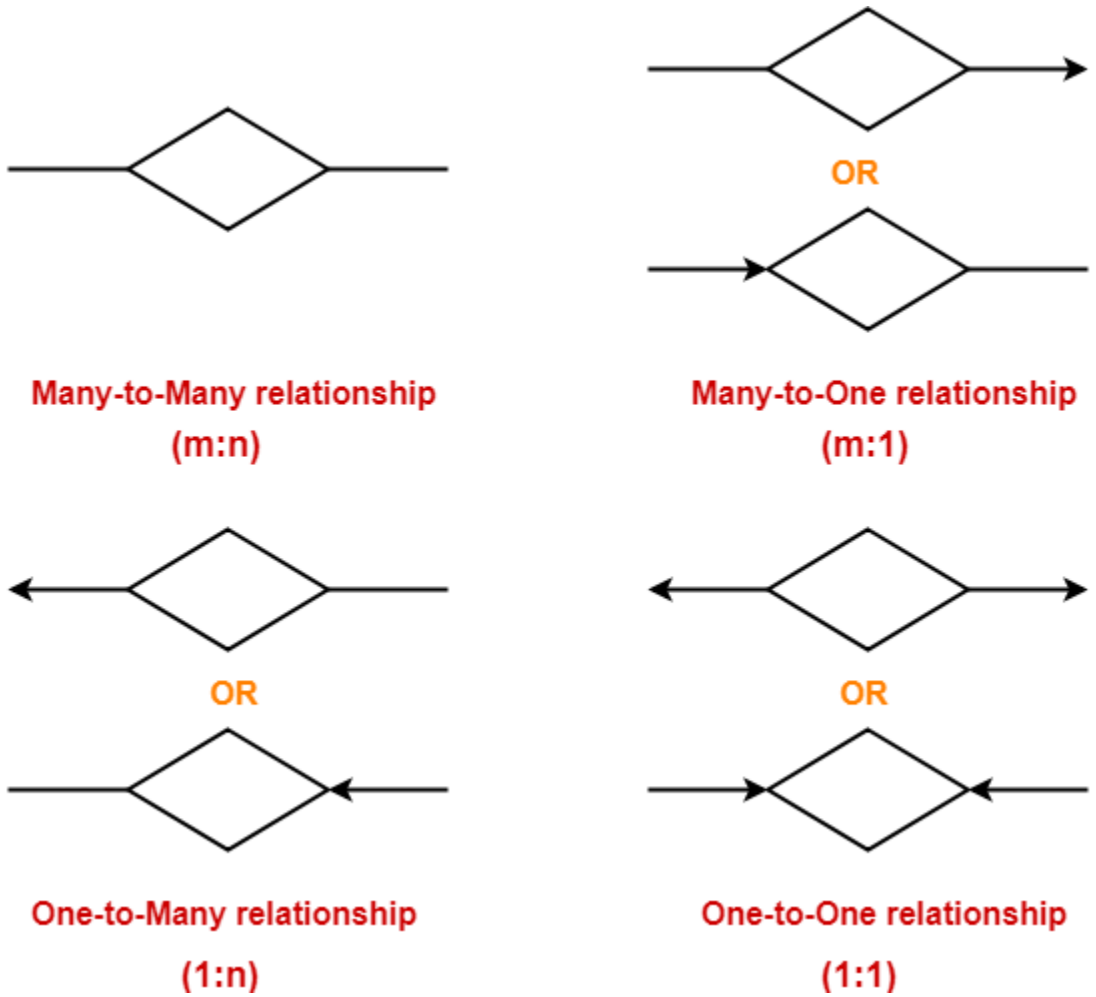
Cardinality constraint defines the maximum number of relationship instances in which an entity can participate.

There are 4 types of cardinality ratios-



1. Many-to-many cardinality (m:n)

2. Many-to-one cardinality (m:1)
3. One-to-many cardinality (1:n)
4. One-to-one cardinality (1:1)

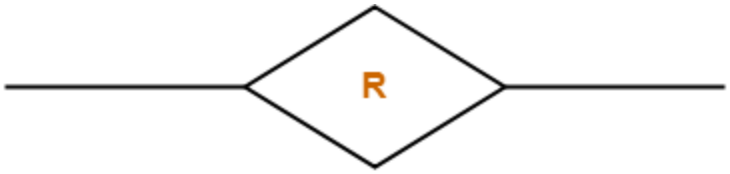


1. Many-to-Many Cardinality-

By this cardinality constraint,

- An entity in set A can be associated with any number (zero or more) of entities in set B.
- An entity in set B can be associated with any number (zero or more) of entities in set A.

Symbol Used-



Cardinality Ratio = m : n

Example-

Consider the following ER diagram-

Many to Many Relationship

Here,

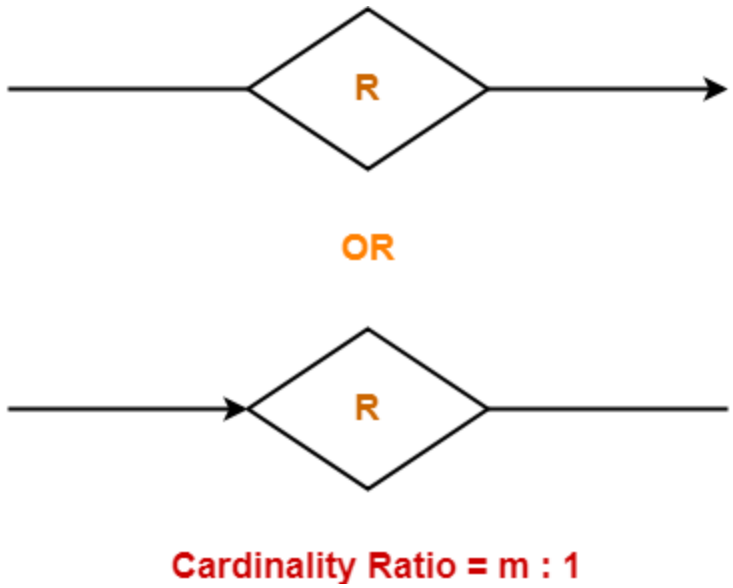
- One student can enroll in any number (zero or more) of courses.
- One course can be enrolled by any number (zero or more) of students.

2. Many-to-One Cardinality-

By this cardinality constraint,

- An entity in set A can be associated with at most one entity in set B.
- An entity in set B can be associated with any number (zero or more) of entities in set A.

Symbol Used-



Example-

Consider the following ER diagram-

Here,

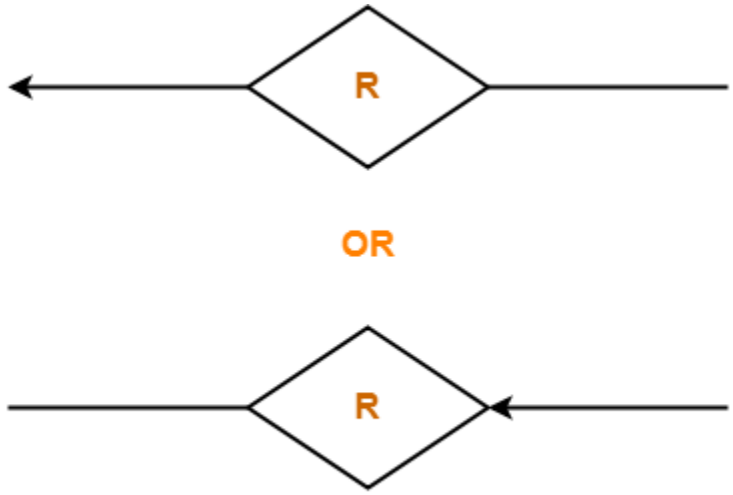
- One student can enroll in at most one course.
- One course can be enrolled by any number (zero or more) of students.

3. One-to-Many Cardinality-

By this cardinality constraint,

- An entity in set A can be associated with any number (zero or more) of entities in set B.
- An entity in set B can be associated with at most one entity in set A.

Symbol Used-



Cardinality Ratio = 1 : n

Example-

Consider the following ER diagram-

One to Many Relationship

Here,

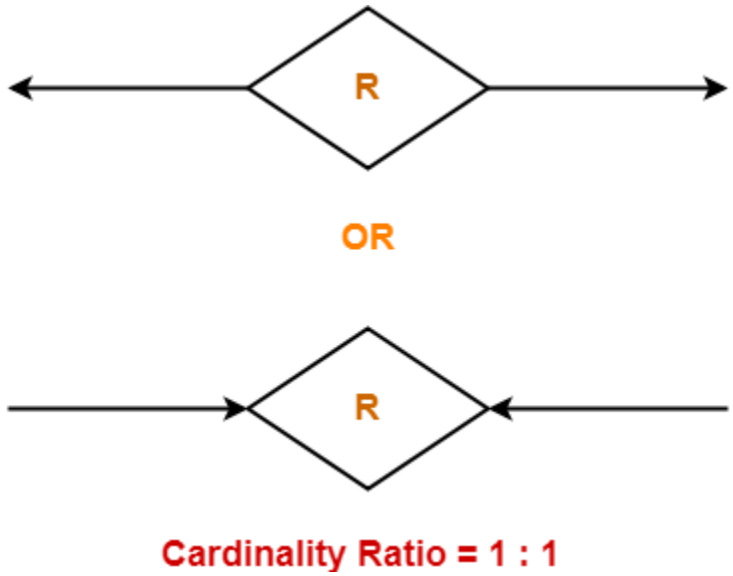
- One student can enroll in any number (zero or more) of courses.
- One course can be enrolled by at most one student.

4. One-to-One Cardinality-

By this cardinality constraint,

- An entity in set A can be associated with at most one entity in set B.
- An entity in set B can be associated with at most one entity in set A.

Symbol Used-



Example-

Consider the following ER diagram-

Here,

- One student can enroll in at most one course.
- One course can be enrolled by at most one student.

Entity Set in DBMS-

- Either a physical existence such as a particular person, office, house or car.
- Or a conceptual existence such as a school, a university, a company or a job.

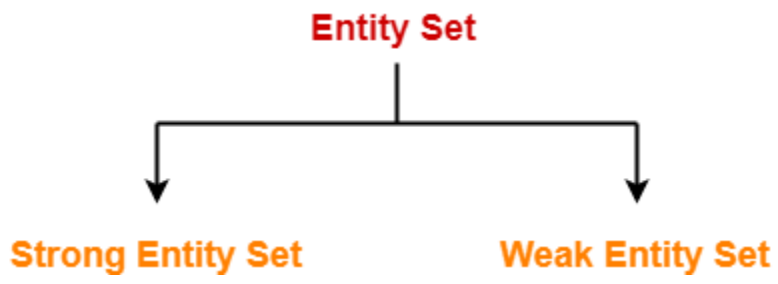
In ER diagram,

- Attributes are associated with an entity set.
- Attributes describe the properties of entities in the entity set.

- Based on the values of certain attributes, an entity can be identified uniquely.

Types of Entity Sets-

An entity set may be of the following two types-



1. Strong entity set
2. Weak entity set

1. Strong Entity Set-

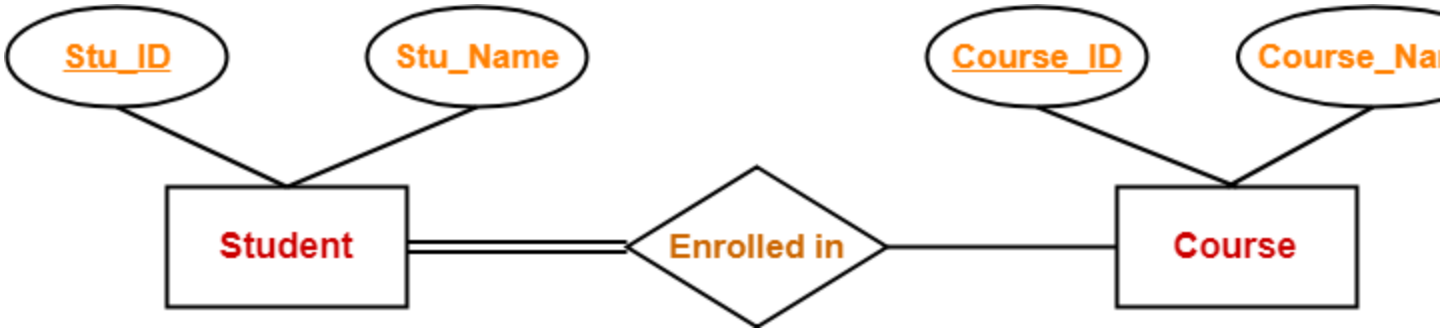
- A strong entity set is an entity set that contains sufficient attributes to uniquely identify all its entities.
- In other words, a primary key exists for a strong entity set.
- Primary key of a strong entity set is represented by underlining it.

Symbols Used-

- A single rectangle is used for representing a strong entity set.
- A diamond symbol is used for representing the relationship that exists between two strong entity sets.
- A single line is used for representing the connection of the strong entity set with the relationship set.
- A double line is used for representing the total participation of an entity set with the relationship set.
- Total participation may or may not exist in the relationship.

Example-

Consider the following ER diagram-



In this ER diagram,

- Two strong entity sets "**Student**" and "**Course**" are related to each other.
- Student ID and Student name are the attributes of entity set "Student".
- Student ID is the primary key using which any student can be identified uniquely.
- Course ID and Course name are the attributes of entity set "Course".
- Course ID is the primary key using which any course can be identified uniquely.
- Double line between Student and relationship set signifies total participation.
- It suggests that each student must be enrolled in at least one course.
- Single line between Course and relationship set signifies partial participation.
- It suggests that there might exist some courses for which no enrollments are made.

2. Weak Entity Set-

- A weak entity set is an entity set that does not contain sufficient attributes to uniquely identify its entities.
- In other words, a primary key does not exist for a weak entity set.
- However, it contains a partial key called as a **discriminator.**
- Discriminator can identify a group of entities from the entity set.
- Discriminator is represented by underlining with a dashed line.

NOTE-

- The combination of discriminator and primary key of the strong entity set makes it possible to uniquely identify all entities of the weak entity set.
- Thus, this combination serves as a primary key for the weak entity set.
- Clearly, this primary key is not formed by the weak entity set completely.

Primary key of weak entity set

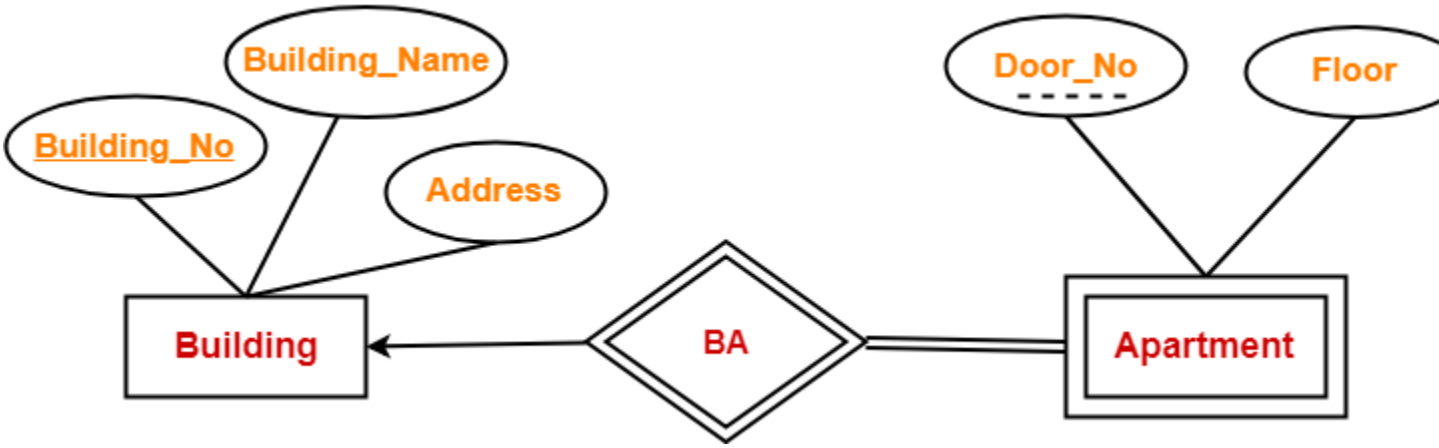
= Its own discriminator + Primary key of strong entity set

Symbols Used-

- A double rectangle is used for representing a weak entity set.
- A double diamond symbol is used for representing the relationship that exists between the strong and weak entity sets and this relationship is known as **identifying relationship**.
- A double line is used for representing the connection of the weak entity set with the relationship set.
- Total participation always exists in the identifying relationship.

Example-

Consider the following ER diagram-



In this ER diagram,

- One strong entity set "**Building**" and one weak entity set "**Apartment**" are related to each other.
- Strong entity set "Building" has building number as its primary key.
- Door number is the discriminator of the weak entity set "Apartment".
- This is because door number alone can not identify an apartment uniquely as there may be several other buildings having the same door number.
- Double line between Apartment and relationship set signifies total participation.
- It suggests that each apartment must be present in at least one building.

- Single line between Building and relationship set signifies partial participation.
- It suggests that there might exist some buildings which has no apartment.

To uniquely identify any apartment,

- First, building number is required to identify the particular building.
- Secondly, door number of the apartment is required to uniquely identify the apartment.

Thus,

Primary key of Apartment

= Primary key of Building + Its own discriminator

= Building number + Door number

<u>Differences between Strong entity set and Weak entity set-</u>

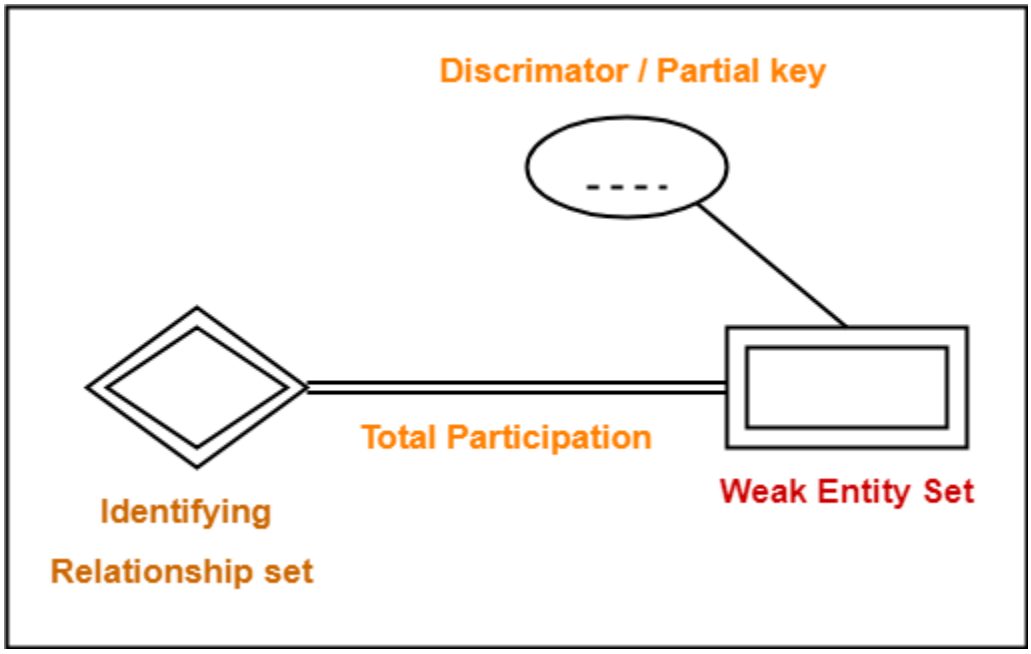
Strong entity set	**Weak entity set**
A single rectangle is used for the representation of a strong entity set.	A double rectangle is used for the representation of a weak entity set.
It contains sufficient attributes to form its primary key.	It does not contain sufficient attributes to form its primary key.

A diamond symbol is used for the representation of the relationship that exists between the two strong entity sets.	A double diamond symbol is used for the representation of the identifying relationship that exists between the strong and weak entity set.
A single line is used for the representation of the connection between the strong entity set and the relationship.	A double line is used for the representation of the connection between the weak entity set and the relationship set.
Total participation may or may not exist in the relationship.	Total participation always exists in the identifying relationship.

Important Note-

In ER diagram, weak entity set is always present in total participation with the identifying relationship set.

So, we always have the picture like shown here-



Relationship in DBMS-

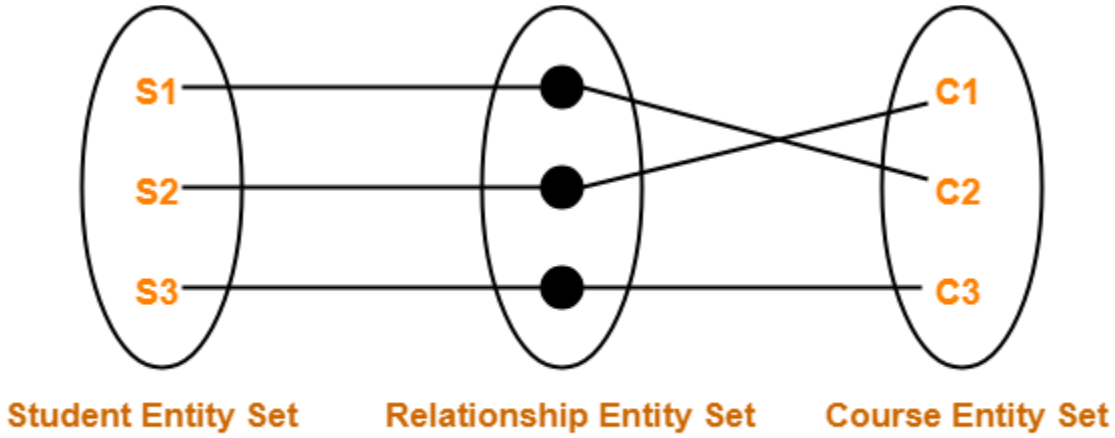
Enrolled in' is a relationship that exists between entities **Student** and **Course**.

Relationship Set-

A relationship set is a set of relationships of same type.

Example-

Set representation of above ER diagram is-



Set Representation of ER Diagram

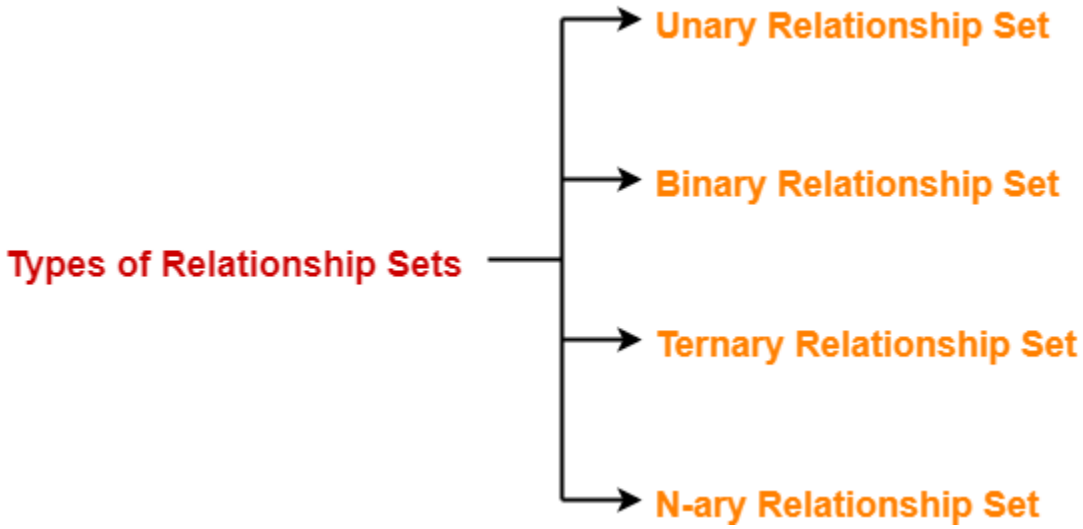
Degree of a Relationship Set-

The number of entity sets that participate in a relationship set is termed as the degree of that relationship set. Thus,

Degree of a relationship set = Number of entity sets participating in a relationship set

Types of Relationship Sets-

On the basis of degree of a relationship set, a relationship set can be classified into the following types-

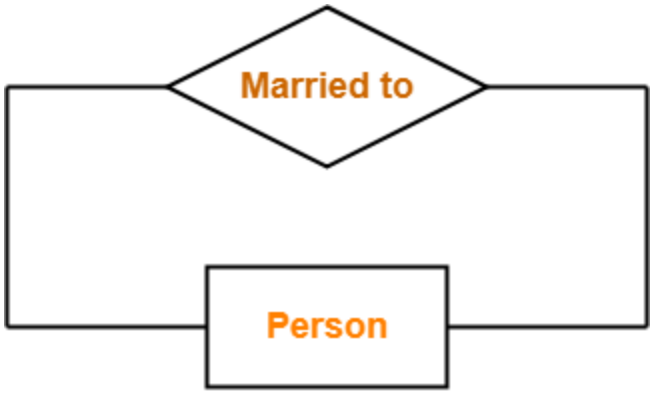


1. Unary relationship set
2. Binary relationship set
3. Ternary relationship set
4. N-ary relationship set

1. Unary Relationship Set-

Unary relationship set is a relationship set where only one entity set participates in a relationship set.

Example-
One person is married to only one person



Unary Relationship Set

2. Binary Relationship Set-

Binary relationship set is a relationship set where two entity sets participate in a relationship set.

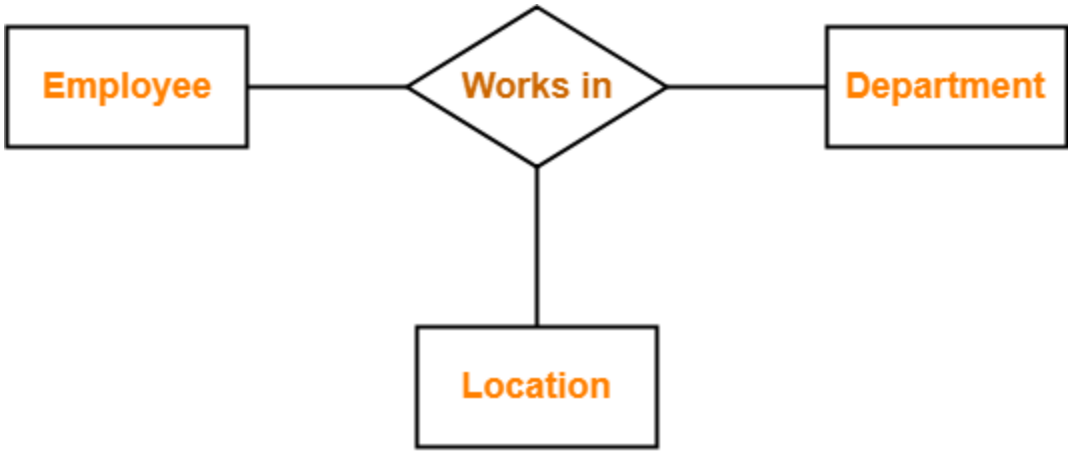
Example-
Student is enrolled in a Course

Binary Relationship Set

3. Ternary Relationship Set-

Ternary relationship set is a relationship set where three entity sets participate in a relationship set.

Example-



Ternary Relationship Set

4. N-ary Relationship Set-

N-ary relationship set is a relationship set where ‘n’ entity sets participate in a relationship set.

Converting ER Diagrams to Tables-

After designing an **ER Diagram**,

- ER diagram is converted into the tables in relational model.
- This is because relational models can be easily implemented by RDBMS like MySQL , Oracle etc.

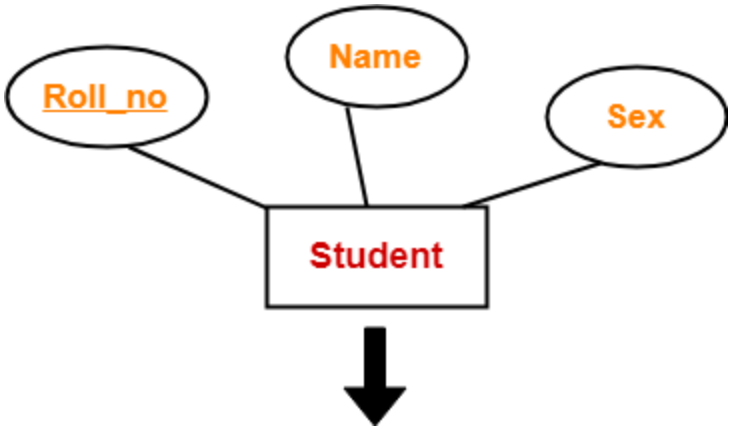
Following rules are used for converting an ER diagram into the tables-

Rule-01: For Strong Entity Set With Only Simple Attributes-

A strong entity set with only simple attributes will require only one table in relational model.

- Attributes of the table will be the attributes of the entity set.
- The primary key of the table will be the key attribute of the entity set.

Example-



Roll_no	**Name**	**Sex**

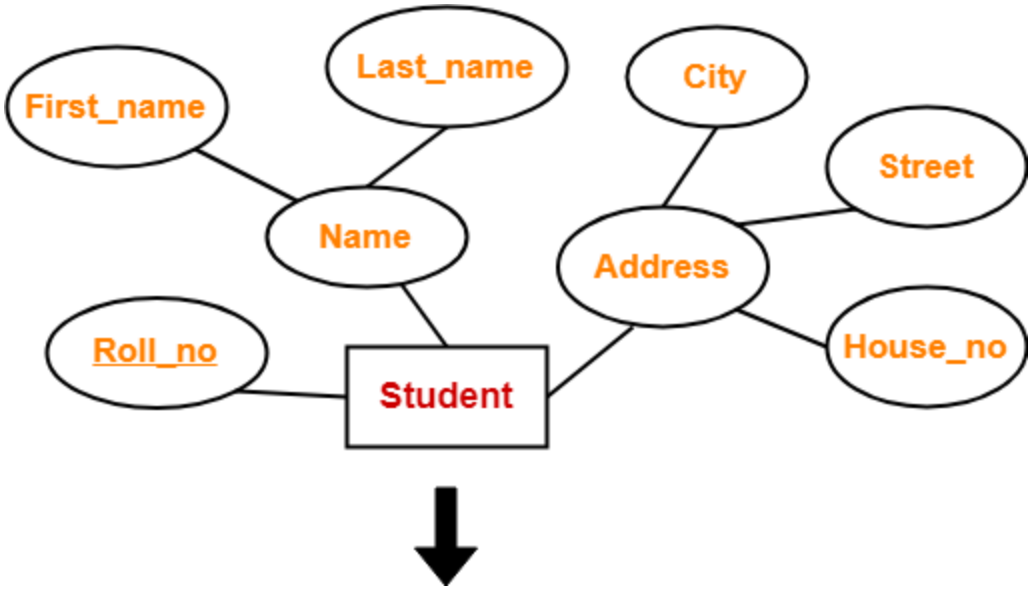
Schema : Student (Roll_no , Name , Sex)

Also Read- Entity Sets in DBMS

Rule-02: For Strong Entity Set With Composite Attributes-

- A strong entity set with any number of composite attributes will require only one table in relational model.
- While conversion, simple attributes of the composite attributes are taken into account and not the composite attribute itself.

Example-



Roll_no	**First_name**	**Last_name**	**House_no**	**Street**	**City**

Schema : Student (Roll_no , First_name , Last_name , House_no , Street , City)

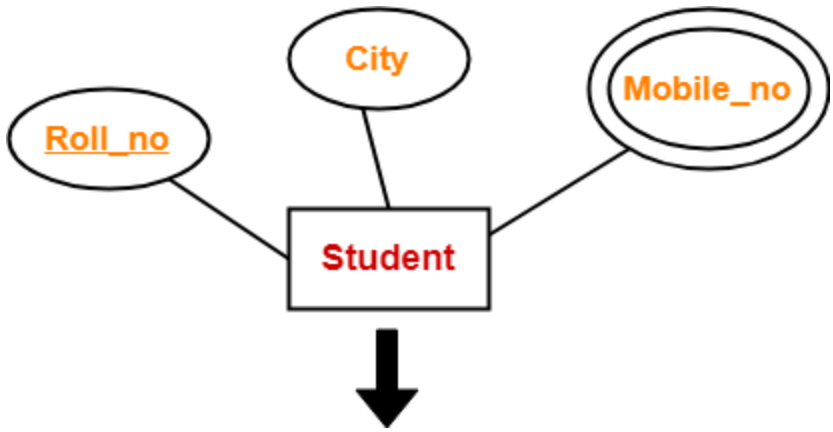
Also Read- Types of Attributes in DBMS

Rule-03: For Strong Entity Set With Multi Valued Attributes-

A strong entity set with any number of multi valued attributes will require two tables in relational model.

- One table will contain all the simple attributes with the primary key.
- Other table will contain the primary key and all the multi valued attributes.

Example-



Roll_no	**City**

Roll_no	**Mobile_no**

Rule-04: Translating Relationship Set into a Table-

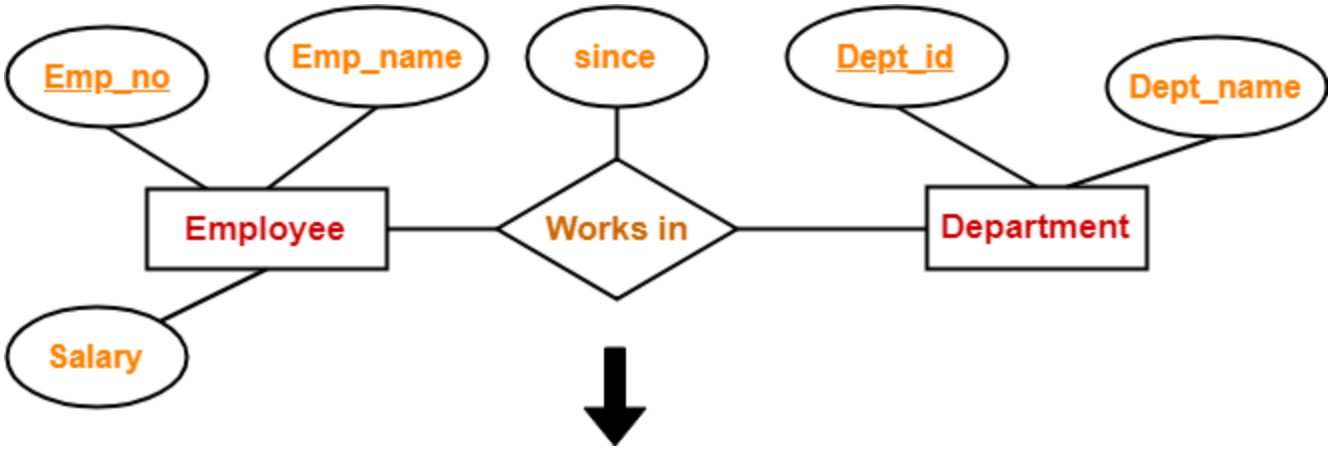
A relationship set will require one table in the relational model.

Attributes of the table are-

- Primary key attributes of the participating entity sets
- Its own descriptive attributes if any.

Set of non-descriptive attributes will be the primary key.

Example-



Emp_no	**Dept_id**	**since**

Schema : Works in (Emp_no , Dept_id , since)

NOTE-

If we consider the overall ER diagram, three tables will be required in relational model-

- One table for the entity set "Employee"
- One table for the entity set "Department"
- One table for the relationship set "Works in"

Rule-05: For Binary Relationships With Cardinality Ratios-

The following four cases are possible-

Case-01: Binary relationship with cardinality ratio m:n

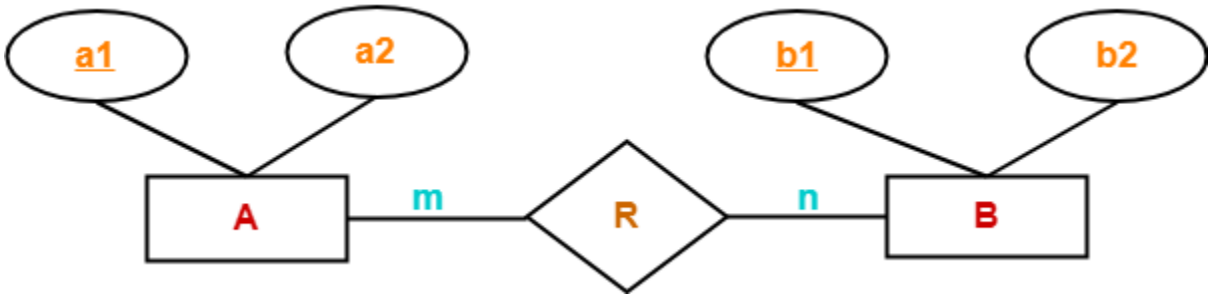
Case-02: Binary relationship with cardinality ratio 1:n

Case-03: Binary relationship with cardinality ratio m:1

Case-04: Binary relationship with cardinality ratio 1:1

Also read- Cardinality Ratios in DBMS

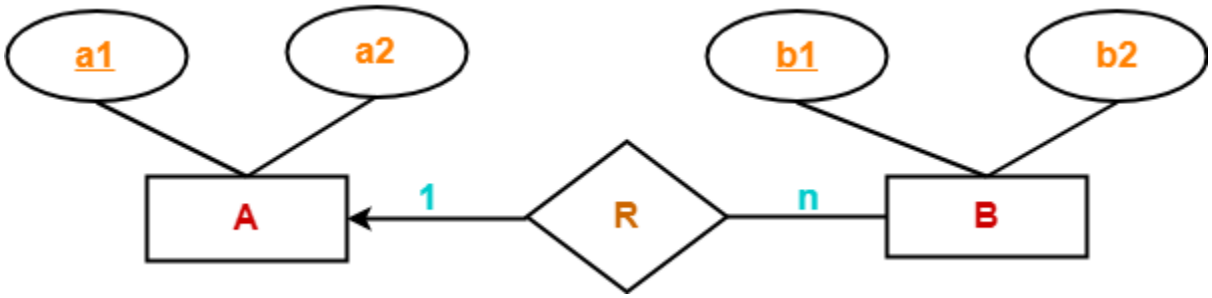
Case-01: For Binary Relationship With Cardinality Ratio m:n



Here, three tables will be required-

1. A (a1 , a2)
2. R (a1 , b1)
3. B (b1 , b2)

Case-02: For Binary Relationship With Cardinality Ratio 1:n

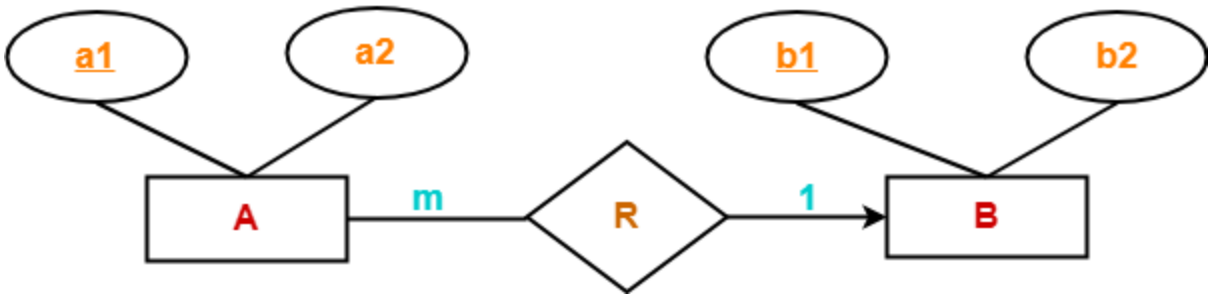


Here, two tables will be required-

1. A (a1 , a2)
2. BR (a1 , b1 , b2)

NOTE- Here, combined table will be drawn for the entity set B and relationship set R.

Case-03: For Binary Relationship With Cardinality Ratio m:1

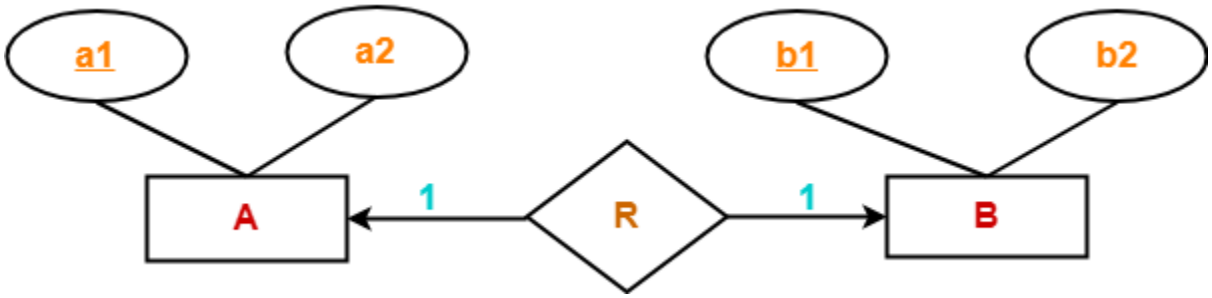


Here, two tables will be required-

1. AR (a1 , a2 , b1)
2. B (b1 , b2)

NOTE- Here, combined table will be drawn for the entity set A and relationship set R.

Case-04: For Binary Relationship With Cardinality Ratio 1:1



Here, two tables will be required. Either combine ‘R’ with ‘A’ or ‘B’

Way-01:

1. AR (a1 , a2 , b1)
2. B (b1 , b2)

Way-02:

1. A (a1 , a2)
2. BR (a1 , b1 , b2)

Thumb Rules to Remember

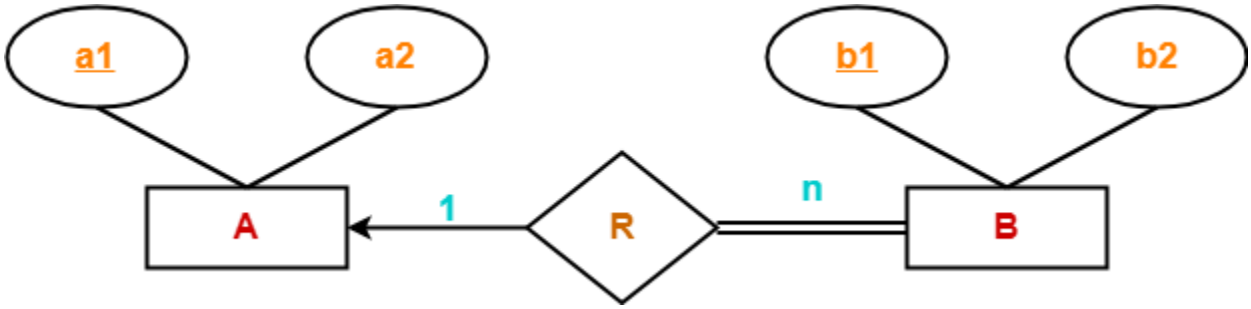
While determining the minimum number of tables required for binary relationships with given cardinality ratios, following thumb rules must be kept in mind-

- For binary relationship with cardinality ration m : n , separate and individual tables will be drawn for each entity set and relationship.
- For binary relationship with cardinality ratio either m : 1 or 1 : n , always remember "many side will consume the relationship" i.e. a combined table will be drawn for many side entity set and relationship set.
- For binary relationship with cardinality ratio 1 : 1 , two tables will be required. You can combine the relationship set with any one of the entity sets.

Rule-06: For Binary Relationship With Both Cardinality Constraints and Participation Constraints-

- Cardinality constraints will be implemented as discussed in Rule-05.
- Because of the total participation constraint, foreign key acquires **NOT NULL** constraint i.e. now foreign key can not be null.

Case-01: For Binary Relationship With Cardinality Constraint and Total Participation Constraint From One Side-

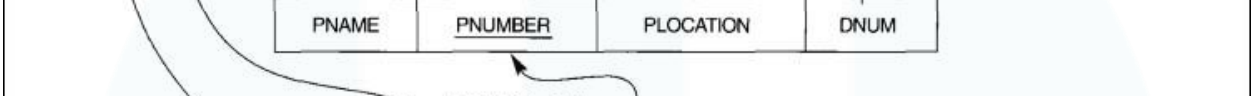


Because cardinality ratio = 1 : n , so we will combine the entity set B and relationship set R.

Then, two tables will be required-

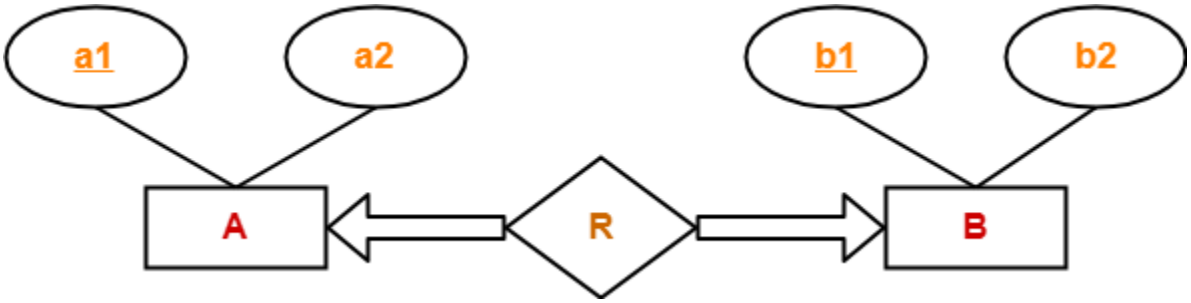
1. A (a1 , a2)
2. BR (a1 , b1 , b2)

Because of total participation, foreign key a1 has acquired NOT NULL constraint, so it can't be null now.



Case-02: For Binary Relationship With Cardinality Constraint and Total Participation Constraint From Both Sides-

If there is a key constraint from both the sides of an entity set with total participation, then that binary relationship is represented using only single table.

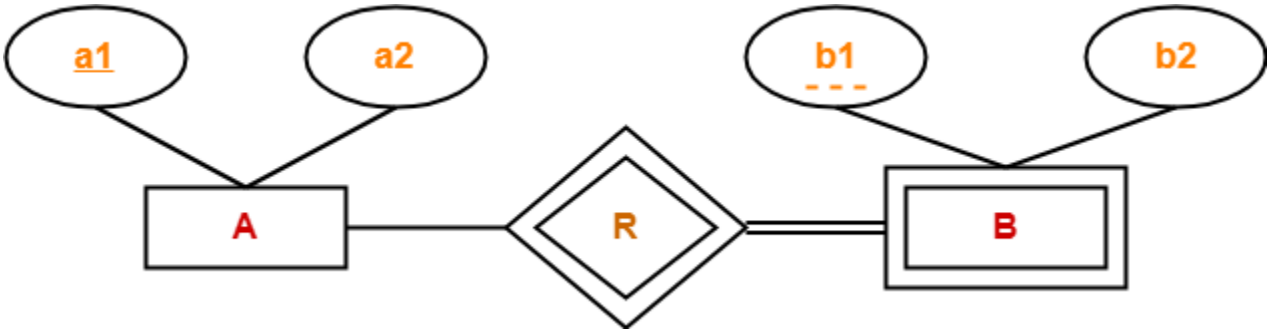


Here, Only one table is required.

- ARB (a1 , a2 , b1 , b2)

Rule-07: For Binary Relationship With Weak Entity Set-

Weak entity set always appears in association with identifying relationship with total participation constraint.



Here, two tables will be required-

1. A (a1 , a2)
2. BR (a1 , b1 , b2)