UNIT - II
INTRODUCTION TO RELATIONAL MODEL

- The relational model is the theoretical basis of relational databases
- The relational model of data is based on the concept of relations
- A "Relation" is a mathematical concept based on the ideas of sets
- The Relational Model was proposed by E.F. Codd for IBM in 1 970 to model data in the form of relations or tables.

WHAT IS RELATIONAL MODEL?

- Relational Model represents how data is stored in Relational Databases. A relational database stores data in the form of relations (tables).
    - After designing the conceptual model of Database using ER diagram, we need to convert the conceptual model in the relational model which can be implemented using any RDMBS languages
    - RDMBS languages: Oracle, SQL, MySQL etc.

WHAT IS RDBMS?

- RDBMS stands for: Relational Database management System. RDBMS is the basis for SQL, and for all modern database systems like MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.
- A Relational database management system (RDBMS) is a database management system (DBMS) that is based on the relational model as introduced by E. F. Codd.
- Current popular RDBMS include:
    - DB2 & Informix Dynamic Server - from I BM
    - Oracle & Rdb - from Oracle
    - SOL Server & MS Access - from Microsoft

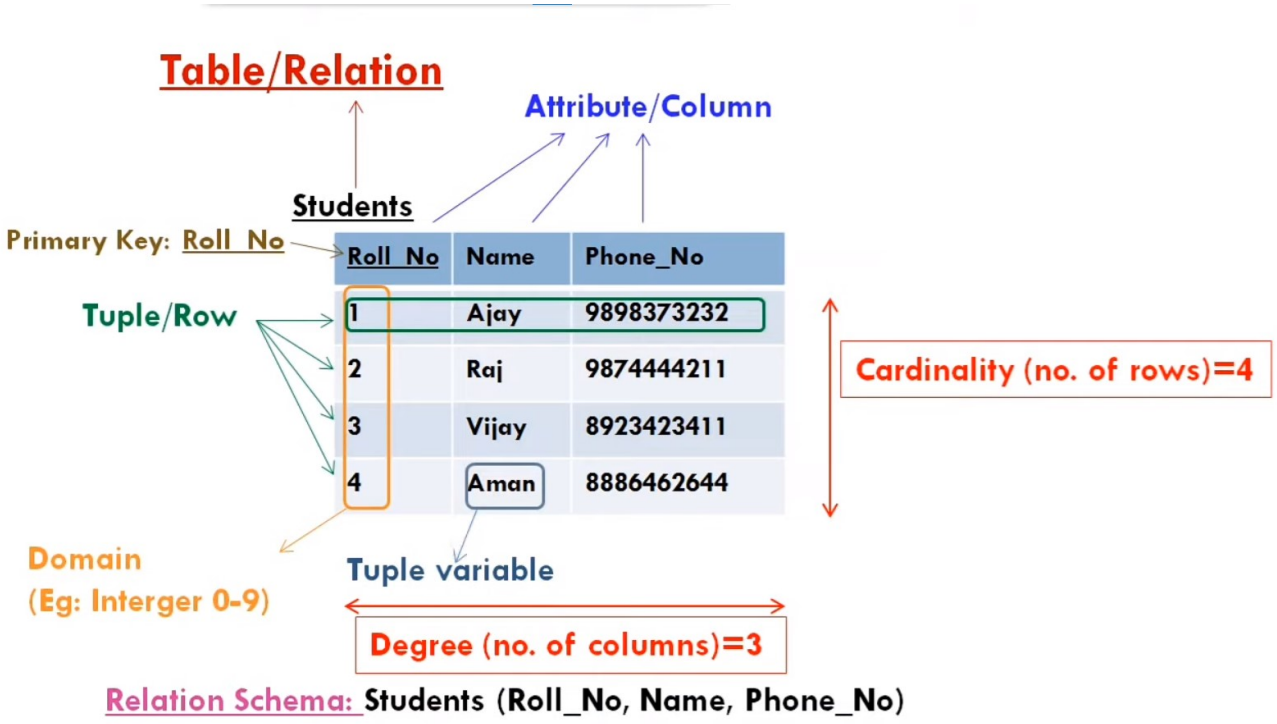
RELATIONAL MODEL CONCEPT

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

Students

R. No	Name	Phone No
1	Alay	9898373232
2	Rai	9874444211
3	Vijay	8923423411
4	Aman	8886462644

Relation: A relation is a table with columns and rows.
Attribute: An attribute is a named column of a relation.
Domain: A domain is the set of allowable values for one or more attributes.
Tuple: A tuple is a row of a relation.
Relation Schema: A relation schema represents the name of the relation with its attributes.
Relation instance (State): Relation instance is a finite set of tuples. Relation instances never have duplicate tuples.
Degree: The total number of columns or attributes in the relation
Cardinality: Total number of rows present in the Table.
Relation key: Every row has one or multiple attributes, that can uniquely identify the row in the relation, which is called relation key (Primary key).
Tuple Variable: it is the data stored in a record of the table



- Each Relation has unique name
- Each tuple/Row is Unique: No duplicate row
- Entries in any column have the same domain.
- Each attribute/column has a unique name
- Order of the columns or rows is irrelevant i.e. relations are unordered
- Each cell of relation contains exactly one value i.e. attribute values are required to be atomic

Relational Model: Mathematical Structure

- The relational model is the theoretical basis of relational databases
- The relational model of data is based on the theory of relations "Relation" is a mathematical concept based on the ideas of sets
- The Relational Model was proposed by E.F. Codd for IBM in 1970 to model data in the form of relations or tables.
- Relational Model represents how data is stored in Relational Databases.
- A relational database stores data in the form of relations (tables).

Mathematical Structure

Relation is a relationship between different types of entities.

Relation is define as a n-ary tuple, its n attributes, al, a2… an. Each attribute ai comes from a domain Di.

It means a particular relation is formed of n attributes, a_l to a_n. Each attribute a_i comes from a domain Di i.e **ai € Di**

So, a relation **r** is a subset of the cross (Cartesian) product of the sets of Di xD2x...x Dn

Mathematical Relation

Suppose we have two sets Dl and D 2,

where D1 = {2,4}

and D2 = {1,3,5}

The Cartesian product of these two sets, written as Dl x D2, is the set of all ordered pairs such that the first element is a member of set Dl and second element is a member of set D2.
Dl x D2 = {(2,1 (2,3), (2,5), (4,1), (4,5),(4,3)}
so r = {(2,1), (4,1)} is a relation over Dl x D2

Relational Model Mathematical Definition

Mathematics define a relation to be a subset of a Cartesian product of a list of domain

Formally, given sets Dl, D2 , ... Dn a relation r is a subset of

Dl xD2x... x-Dn

Thus, a relation is a set of n-tuples (al , a2, an) where ai € Di

For example if al: customer-name = {Jones, Smith, Curry, Lindsay}
a2: customer-street = {Main, North, Park}
a3: customer-city = {Harrison, Rye, Pittsfield}
Then r = {(Jones, Main, Harrison), →1st tuple
(Smith, North, Rye), →2nd tuple
(Curry, North, Rye), →3rd tuple
(Lindsay, Park, Pittsfield) } →4th tuple

is a relation customer-name X customer-street X customer-city

Relations are unordered

- In relational model relation is depicted as table In relation (or table),
- Order of tuples is irrelevant (tuples may be stored in an arbitrary order)
  - i.e. Relation are unordered

r = { (Jones, Main, Harrison),
(Smith, North, Rye),
(Curry, North, Rye),
(Lindsay, Park, Pittsfield) }

customer-name	customer-street	customer-city
Jones	Main	Harrison
Smith	North	Rye
Curry	North	Rye
Lindsay	Park	Pittsfield

Attribute Types

- **Name:** Each attribute of a relation has a name
- **Domain:** The set of allowed values for each attribute is called the domain of the attribute
  - Roll_No: Numeric (of 10 digit)
  - Name: Character (of 1-30)
  - DOB: Date ddmmyyyy
- **Atomic:** Attribute values are required to be atomic, that is, indivisible atomic means cannot be subdivided any further)
  - E.g. multivalued attribute values are not atomic
  - E.g. composite attribute values are not atomic
- **NULL:** The special value null is a member of every domain
  - null value can be used for unknown values or not applicable (NÄ) values
  - The null value causes complications in the definition of many operations

Relational schema

- Schema means that this is the description of the relation
- A1, A2, …An are attributes
- R= (A1,A2,…An) is a relation schema
    - E.g. Customer-schema = (customer-name, customer-street, customer-city)
- A relation is defined over a schema i.e. r(R)
- r(R) is a relation r on the relation schema R
    - E.g. customer (Customer-schema)

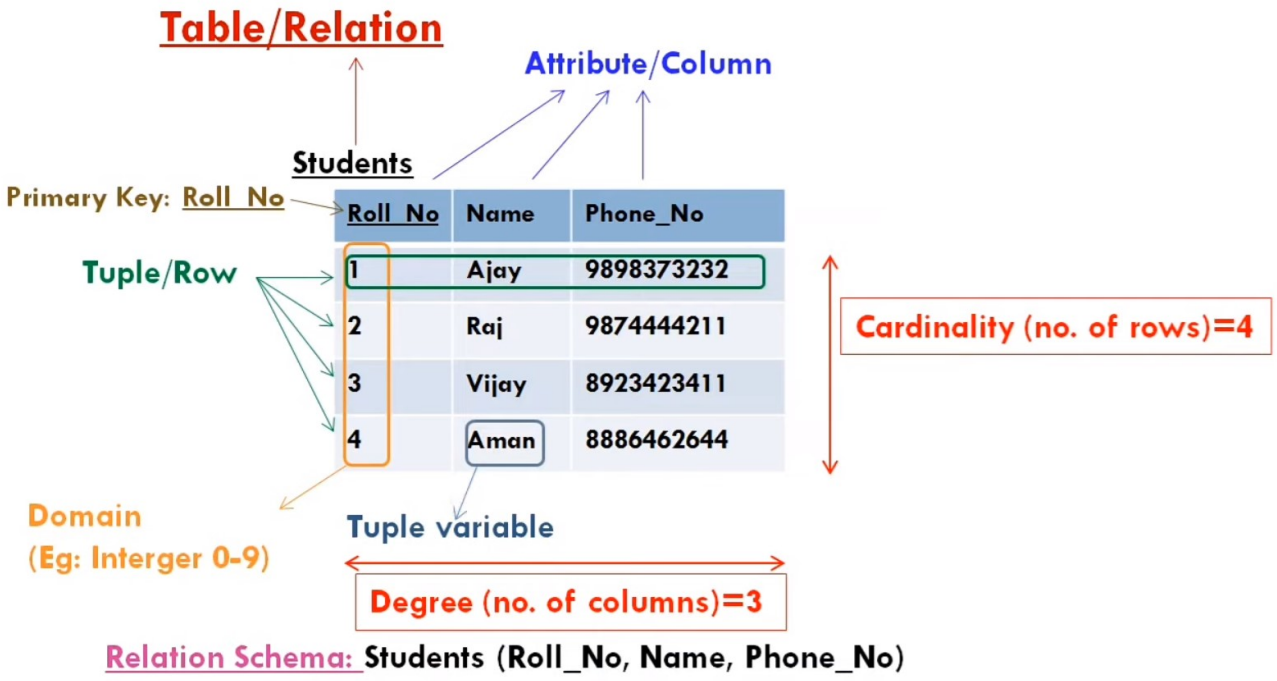
Relation Instance

- The current values (relation instance) of a relation are specified by a table

r = {(Jones, Main, Harrison), →1st tuple
(Smith, North, Rye), →2nd tuple
(Curry, North, Rye), →3rd tuple
(Lindsay, Park, Pittsfield) } →4th tuple

- An element t of r is a tuple, represented by a row in a table
- Attribute Ai represents columns in a table

"Database" is a multiple interrelated relations



Integrity Constraints over Relation

- Integrity constraints are used to ensure accuracy and consistency of the data in a relational database.
- Integrity constraints are of rules that the database is not permitted to violate.
- Constraints may apply to each attribute or they may apply to relationships between tables.
- Integrity constraints ensure that changes (update, deletion, insertion) made to the database by authorized Users do not result in a loss of data consistency. Thus, integrity constraints guard against accidental damage to the database.

- Example - A blood group must be 'A' or 'B' or 'AB' or 'O' only (cannot any other else).

TYPES OF INTEGRITY CONSTRAIN

1. Domain Constraint
2. Entity Integrity Constraint
3. Referential Integrity Constraint
4. Key Constraints

Domain Constraint

- Domain constraints defines the domain or the 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.

STUDENT_ID	NAME	SEMESTER	AGE
101	Manish	1st	18
102	Rohit	3rd	19
103	Badal	5th	20
104	Amit	7th	A

Not allowed. Because AGE is an integer value

Entity Integrity Constraint

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.

EMP_ID	EMP_NAME	SALARY
111	Mohan	20000
112	Rohan	30000
113	Sohan	35000
	Logan	20000

Not allowed as Primary Key can't contain NULL value

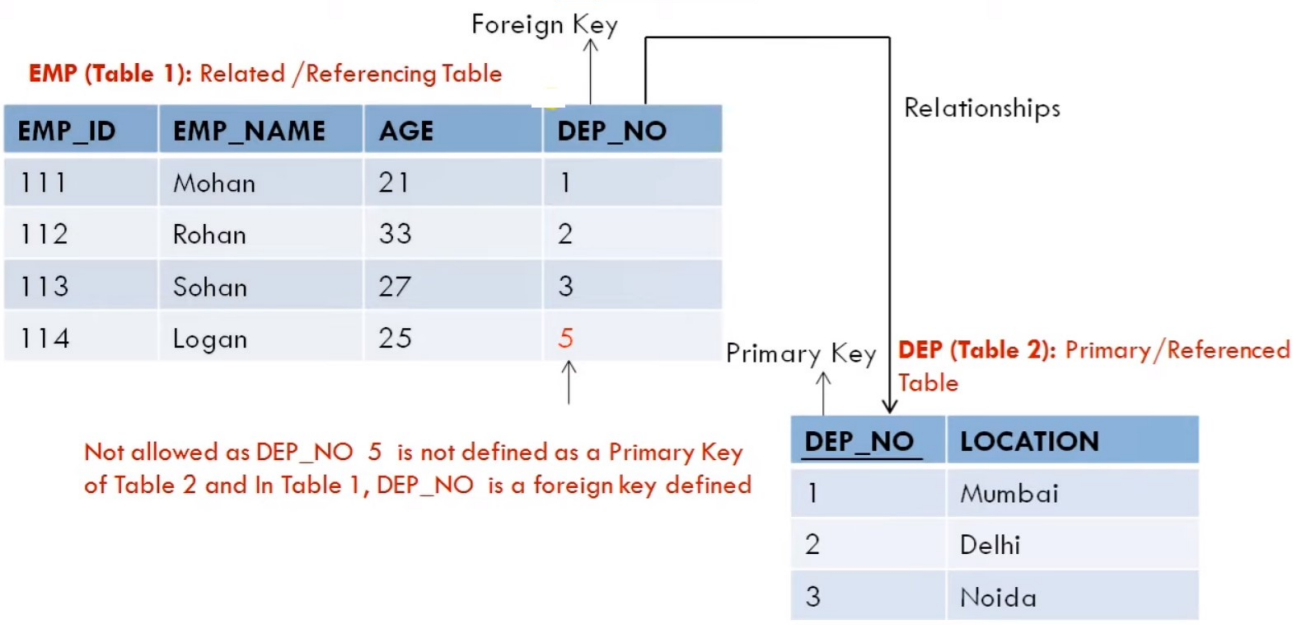
Referential Integrity Constraint

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

The rules are:

- You can't delete a record from a primary table if matching records exist in a related table.
- You can't change a primary key value in the primary table if that record has related records.

- You can't insert a value in the foreign key field of the related table that doesn't exist in the primary key of the primary table.
- However, you can enter a Null value in the foreign key, specifying that the records are unrelated.



Key Constraints

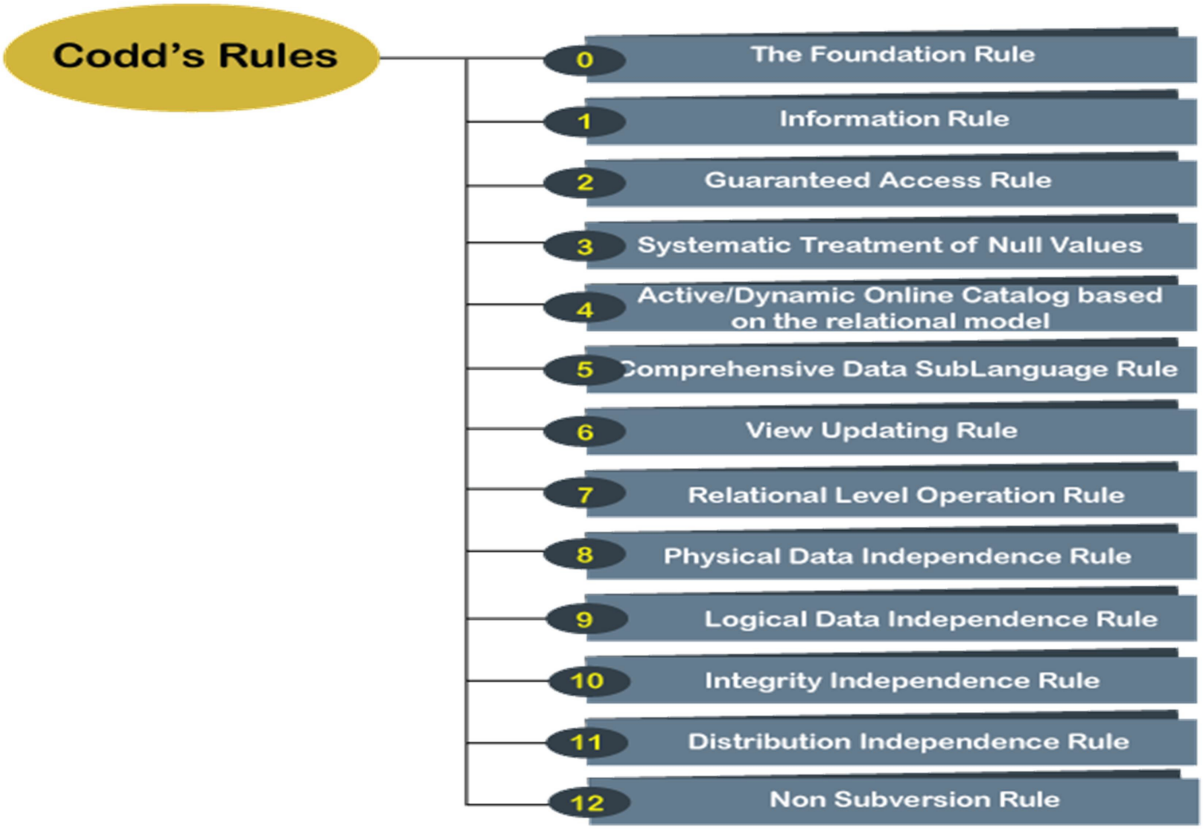
- An entity set can have multiple keys or candidate keys (minimal super key), but out of which one key will be the primary key.
- Key constraint specifies that in any relation-
  - All the values of primary key must be unique.
  - The value of primary key must not be null.

STUDENT_ID	NAME	SEMESTER	AGE
101	Manish	1st	18
102	Rohit	3rd	19
103	Badal	5th	20
102	Amit	7th	21

Not allowed. Because all rows must be **unique**

12 Codd's Rules / Commandments

Every database has tables, and constraints cannot be referred to as a rational database system. And if any database has only relational data model, it cannot be a Relational Database System (RDBMS). So, some rules define a database to be the correct RDBMS. These rules were developed by Dr. Edgar F. Codd (E.F. Codd) in 1970s, who has vast research knowledge on the Relational Model of database Systems. Codd presents his 13 rules for a database to test the concept of DBMS.



Rule 0: The Foundation Rule

- The database must be able to manage data in relational form.

Rule 1: Information Rule

- All data stored in the database must exist as a value of some table cell.

Rule 2: Guaranteed Access Rule

- Every single or precise data (atomic value) may be accessed logically from a relational database using the combination of primary key value, table name, and column name.

Rule 3: Systematic Treatment of Null Values

- Database must support NULL values.

Rule 4: Active/Dynamic Online Catalog based on the relational model

- It represents the entire logical structure of the descriptive database that must be stored online and is known as a database dictionary. It authorizes users to access the database and implement a similar query language to access the database.

Rule 5: Comprehensive Data Sub Language Rule

- Database must support at least one language that supports: data definition, view definition, data manipulation, integrity constraints, authorization, and transaction boundaries. If the database allows access to the data without any language, it is considered a violation of the database.

Rule 6: View Updating Rule

- All views table can be theoretically updated and must be practically updated by the database systems.

Rule 7: Relational Level Operation (High-Level Insert, Update and delete) Rule

- A database system should follow high-level relational operations such as insert, update, and delete in each level or a single row. It also supports union, intersection and minus operation in the database system.

Rule 8: Physical Data Independence Rule

- Data stored in the database must be independent of the applications that can access it i.e., the data stored in the database must not depend on any other data or an application.

Rule 9: Logical Data Independence Rule

- Any change in the logical representation of the data (structure of the tables) must not affect the user's view.

Rule 10: Integrity Independence Rule

- Changing the integrity constraints at the database level should not reflect any change at the application level.

Rule 11: Distribution Independence Rule

- The database must work properly even if the data is stored in multiple locations or is being used by multiple end-users.

Rule 12: Non Subversion Rule

- The non-subversion rule defines RDBMS as a SQL language to store and manipulate the data in the database.
- If a system has a low-level or separate language other than SQL to access the database system, it should not subvert or bypass integrity to transform data.

Logical Database Design (ER Diagram to Relational Model)

After designing an ER Diagram,

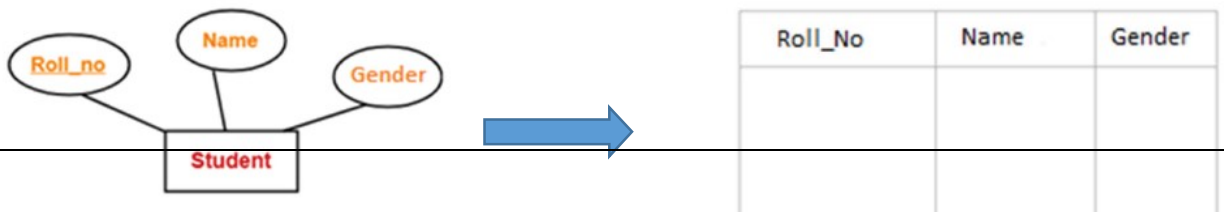
- ER diagram is converted into the tables in relational model.
- This is because relational models can be easily implemented by RDBMS like MySQL, Oracle etc.

In general conversion of E-R diagram into a relational model involves the following:

- Mapping of an entity set into relation (tables) of the database.
- The attributes of a table include the attributes of an entity
- The key attribute of an entity becomes the primary key of the relation

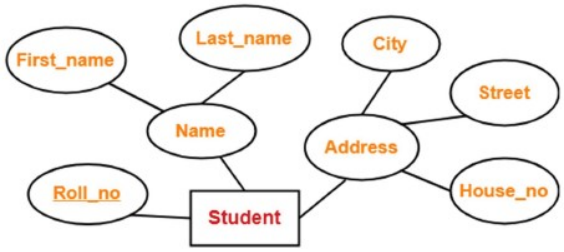
Rule-01: For Strong Entity Set With Only Simple Attributes-

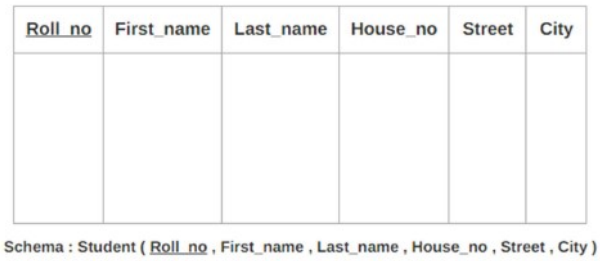
- A strong entity set with only simple attributes will require only one table in relational model.
  - Attributes of the table will be the attributes of the entity set.
  - The primary key of the table will be the key attribute of the entity set.
- Example:



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-**



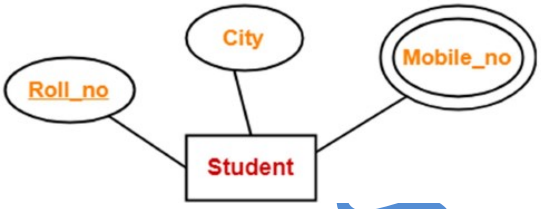


Schema : Student (Roll_no , First_name , Last_name , House_no , Street , City)

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

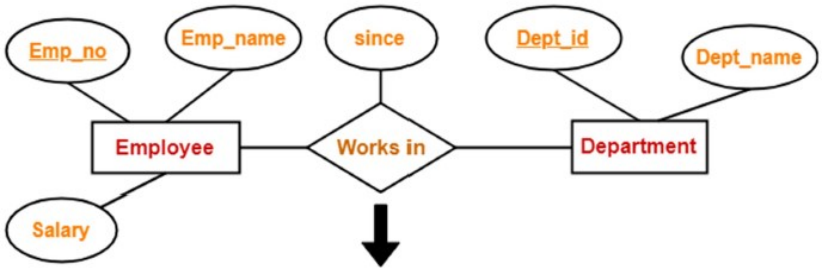
- A strong entity set with any number of multi valued attributes will require two tables in relational model.
    - One table will contain all the simple attributes with the primary key.
    - Other table will contain the primary key and all the multi valued attributes.

Example



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.
- 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”



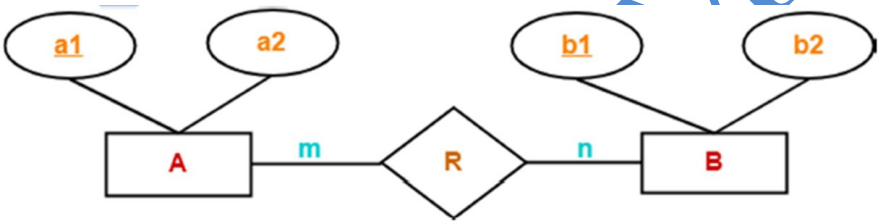
Emp_no	Dept_id	since

Schema : Works in (Emp_no , Dept_id , since)

- **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

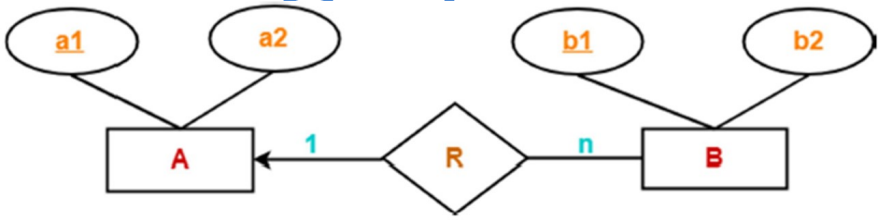
Case-01: Binary relationship with cardinality ratio m:n

Here, three tables will be required-



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

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

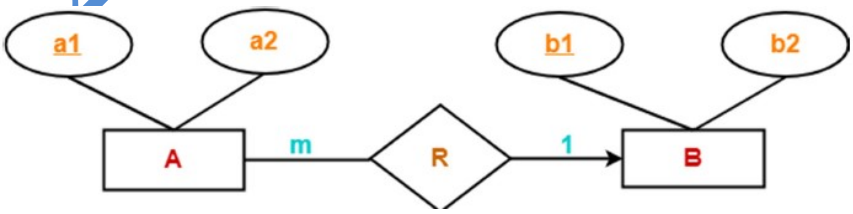


Here, three tables will be required-

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: Binary relationship with cardinality ratio m:1

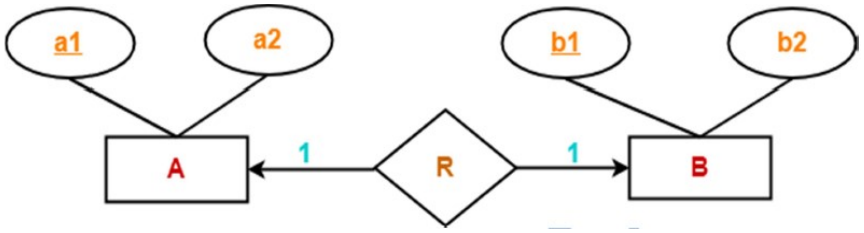


Here, two tables will be required-

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: 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)

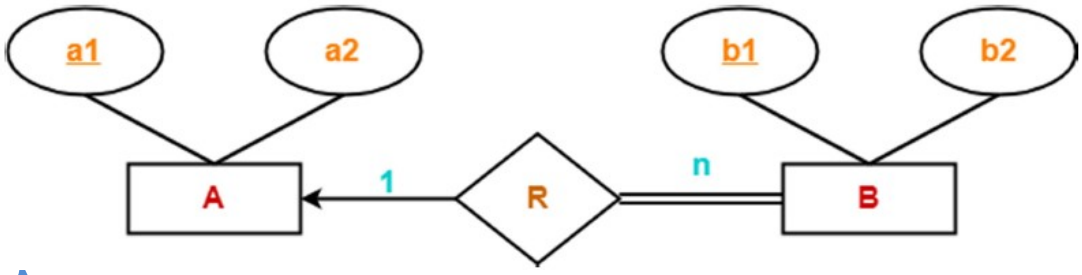
Thumb Rules to Remember

- 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't 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-

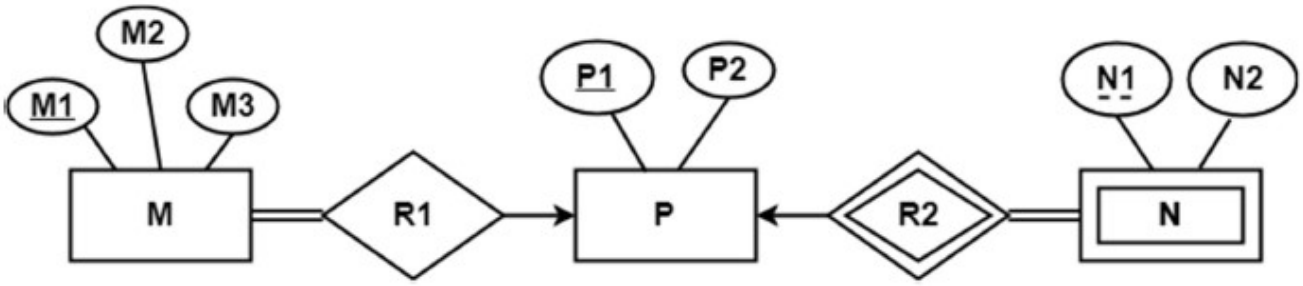
1. A (a1 , a2)
2. BR (a1 , b1 , b2)

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

Problem-01:

Find the minimum number of tables required for the following ER diagram in relational model-

- Applying the rules, minimum 3 tables will be required-



- MR1 (M1 , M2 , M3 , P1)
- P (P1 , P2)
- NR2 (P1 , N1 , N2)