ऋते ज्ञानान्न मुक्तिः

GLA
UNIVERSITY
MATHURA
Recognised by UGC Under Section 2(f)

Accredited with A Grade by NAAC

12-B Status from UGC

BCSC0053
ADVANCED DATABASE MANAGEMENT SYSTEM

Lecture- 1

Presented by:

Atul Kumar Uttam

Assistant Professor

Computer Engineering & Applications Department,
GLA University, Mathura

atul.uttam@gla.ac.in, +91-8979593001

Syllabus: Module 1

Introduction:

- An Overview of Database Management System,
- Database System vs File System,
- Database System Concept and Architecture,
- Data Model
- Schema and Instances,
- Data Independence,
- Database Language and Interfaces (DDL, DML, DCL),
- Database Development Life Cycle (DDLC) with case studies.

Syllabus: Module 1

Data Modeling Using the Entity-Relationship Model:

- ER Model Concepts,
- Notation for ER Diagram,
- Mapping Constraints,
- Keys,
- Specialization, Generalization, Aggregation,
- Reduction of an ER Diagram to Tables.

Relational Data Model and Language:

- Relational Data Model Concepts,
- Integrity Constraints, Entity Integrity, Referential Integrity, Keys Constraints, Domain Constraints,
- Relational Algebra

Syllabus: Module 1

Database Design & Normalization:

- Functional Dependencies,
- Primary Key, Foreign Key, Candidate Key, Super Key, Normal Forms,
- First, Second, Third Normal Forms, BCNF, 4th Normal Form, 5th Normal Form,
- Lossless Join Decompositions, Non Redundant Cover, Canonical Cover, MVD and JDs, Inclusion Dependence.

Syllabus: Module 2

Transaction Processing Concept:

- Transaction System,
- Testing of Serializability,
- Serializability of Schedules,
- Conflict & View Serializable Schedule, Recoverability,
- Recovery from Transaction Failures,
- Log Based Recovery,
- Deadlock Handling.

Concurrency Control Techniques:

- Concurrency Control,
- Locking Techniques for Concurrency Control, 2PL,
- Time Stamping Protocols for Concurrency Control,
- Validation Based Protocol.

Syllabus: Module 2

Distributed Database:

- Introduction of Distributed Database,
- Data Fragmentation and Replication.

NoSQL System:

- RDBMS vs NoSQL,
- BASE properties, Key-value, Columnar,
- Document and Graph-Based database,
- Introduction of MongoDB, Cassandra, Neo4j and Risk.

Database Programming using Python:

- Database connectivity, Retrieving Data from Database,
- Parameters Passing, Execute many Methods,
- Cursor Attributes, Invoke Stored Procedures, Invoke Stored Functions.

Books to Follow

Text Book:

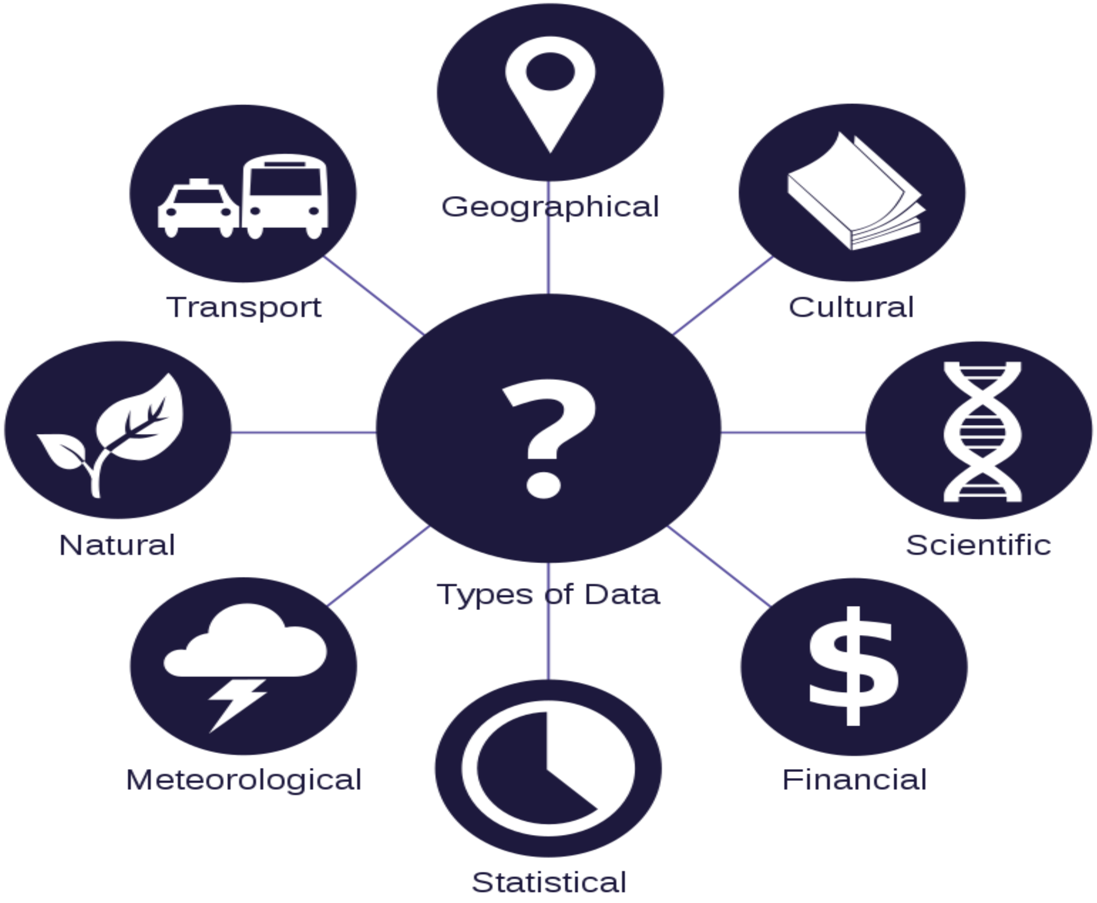
- Fundamentals of Database Systems, 7th Edition, Ramez Elmasri, Shamkant B. Navathe, 2016 Pearson.
- Sadalage, P. & Fowler , “NoSQL Distilled: A Brief Guide to the Emerging World of Polyglot Persistence”, Pearson Education, 2012.

Reference Books:

- Date C J,” An Introduction to Database Systems”, 8th Edition, Addison Wesley.
- Database System Concepts 7th Edition Avi Silberschatz Henry F. Korth, S. Sudarshan 2019 McGraw-Hill.
- Redmond, E. &Wilson, “Seven Databases in Seven Weeks: A Guide to Modern Databases and the NoSQL Movement”, 1st Edition.

Data

- Known facts that can be recorded and that have implicit meaning.



https://en.wikipedia.org/wiki/Data

Definitions

- **Data:**
  - Known facts that can be recorded and have an implicit meaning.
- **Database:**
  - A collection of related data.
- **Database Management System (DBMS):**

  DBMS contains information about a particular enterprise
  - Collection of interrelated data
  - Set of programs to access the data
  - An environment that is both *convenient* and *efficient* to use
- **Mini-world:**
  - Some part of the real world about which data is stored in a database. For example, student grades and transcripts at a university.

University Database Example

- Application program examples
  - Add new students, instructors, and courses
  - Register students for courses, and generate class rosters
  - Assign grades to students, compute grade point averages (GPA) and generate transcripts

- In the early days, database applications were built directly on top of **file systems**

Database Management System(DBMS)

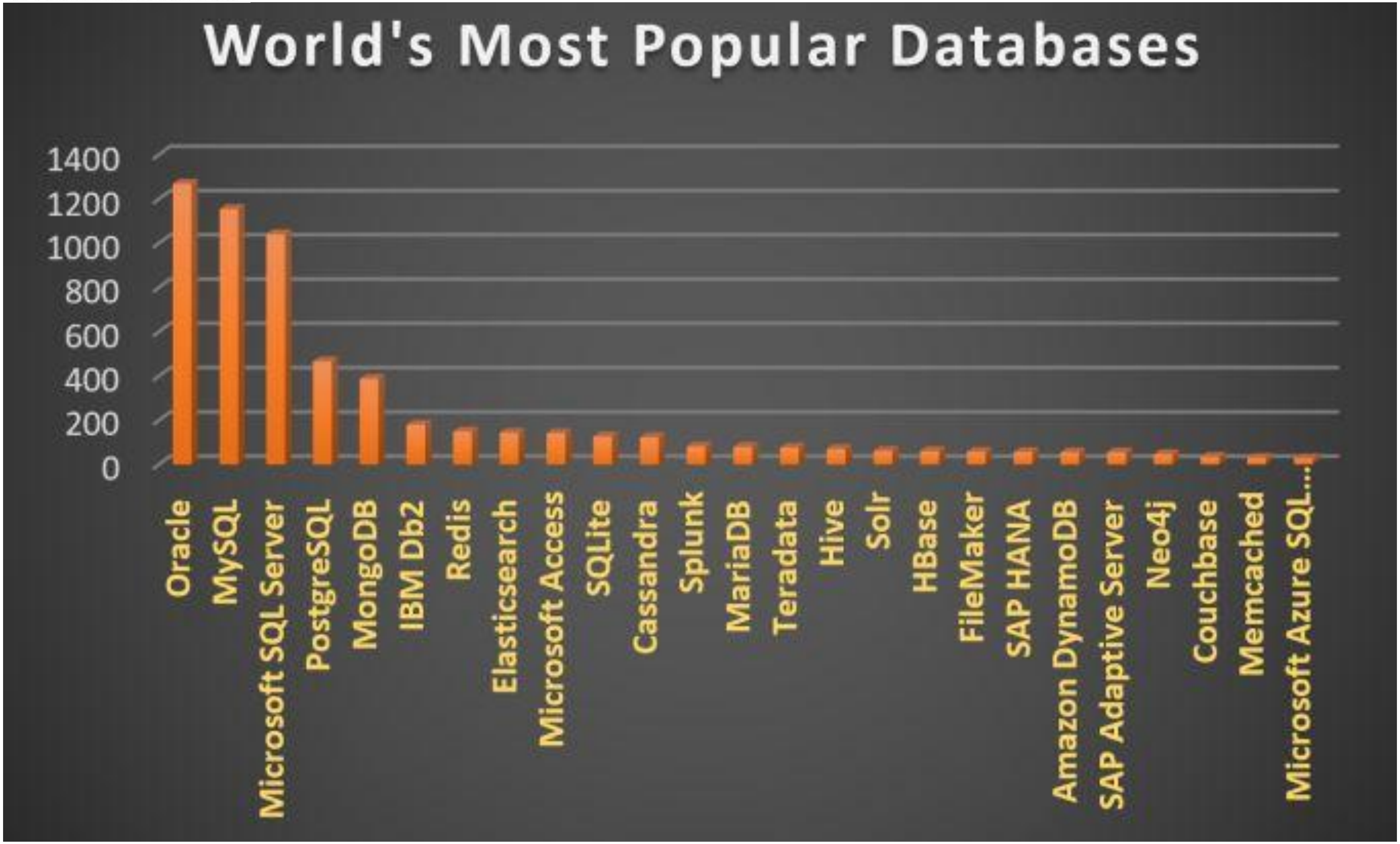
- **DBMS** is a collection of programs that enables users to create and maintain a database.
- The primary goal of a DBMS is to provide a way to store and retrieve database information that is both convenient and efficient.
- **The DBMS is hence a general-purpose software system that facilitates the processes of defining, constructing , manipulating and sharing databases among various users and application.**

- **Defining :** specifying the data types, structures, and constraints for the data.

- **Constructing :**includes storing the data itself on some storage medium.

- **Manipulating :**includes querying the database to retrieve specific data, updating the data to reflect changes in data, generating reports from data.

- **Sharing :** allows multiple users to access the database concurrently.


World's Most Popular Databases

https://www.c-sharpcorner.com/article/what-is-the-most-popular-database-in-the-world/

Drawbacks of using file systems to store data

- **Data redundancy and inconsistency**
  - Multiple file formats, duplication of information in different files
- **Difficulty in accessing data**
  - Need to write a new program to carry out each new task
- **Data isolation**
  - Multiple files and formats
- **Integrity problems**
  - Integrity constraints (e.g., account balance > 0) become “buried” in program code rather than being stated explicitly
  - Hard to add new constraints or change existing ones

Drawbacks of using file systems to store data

- **Atomicity of updates**
  - Failures may leave database in an inconsistent state with partial updates carried out
  - Example: Transfer of funds from one account to another should either complete or not happen at all
- **Concurrent access by multiple users**
  - Concurrent access needed for performance
  - Uncontrolled concurrent accesses can lead to inconsistencies
    - Example: Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) at the same time
- **Security problems**
  - Hard to provide user access to some, but not all, data

Database systems offer solutions to all the above problems

DBMS Disadvantages

- DBMS may involve unnecessary overhead costs that would not be incurred in traditional file processing.

- The overhead costs of using a DBMS are due to the following:
  - High initial investment in hardware, software, and training
  - The generality that a DBMS provides for defining and processing data
  - Overhead for providing security, concurrency control, recovery, and integrity functions

When not to use a DBMS

- **Main inhibitors (costs) of using a DBMS:**
  - High initial investment and possible need for additional hardware.
  - Overhead for providing generality, security, concurrency control, recovery, and integrity functions.
- **When a DBMS may be unnecessary:**
  - If the database and applications are simple, well defined, and not expected to change.
  - If there are stringent real-time requirements that may not be met because of DBMS overhead.
  - If access to data by multiple users is not required.

When not to use a DBMS

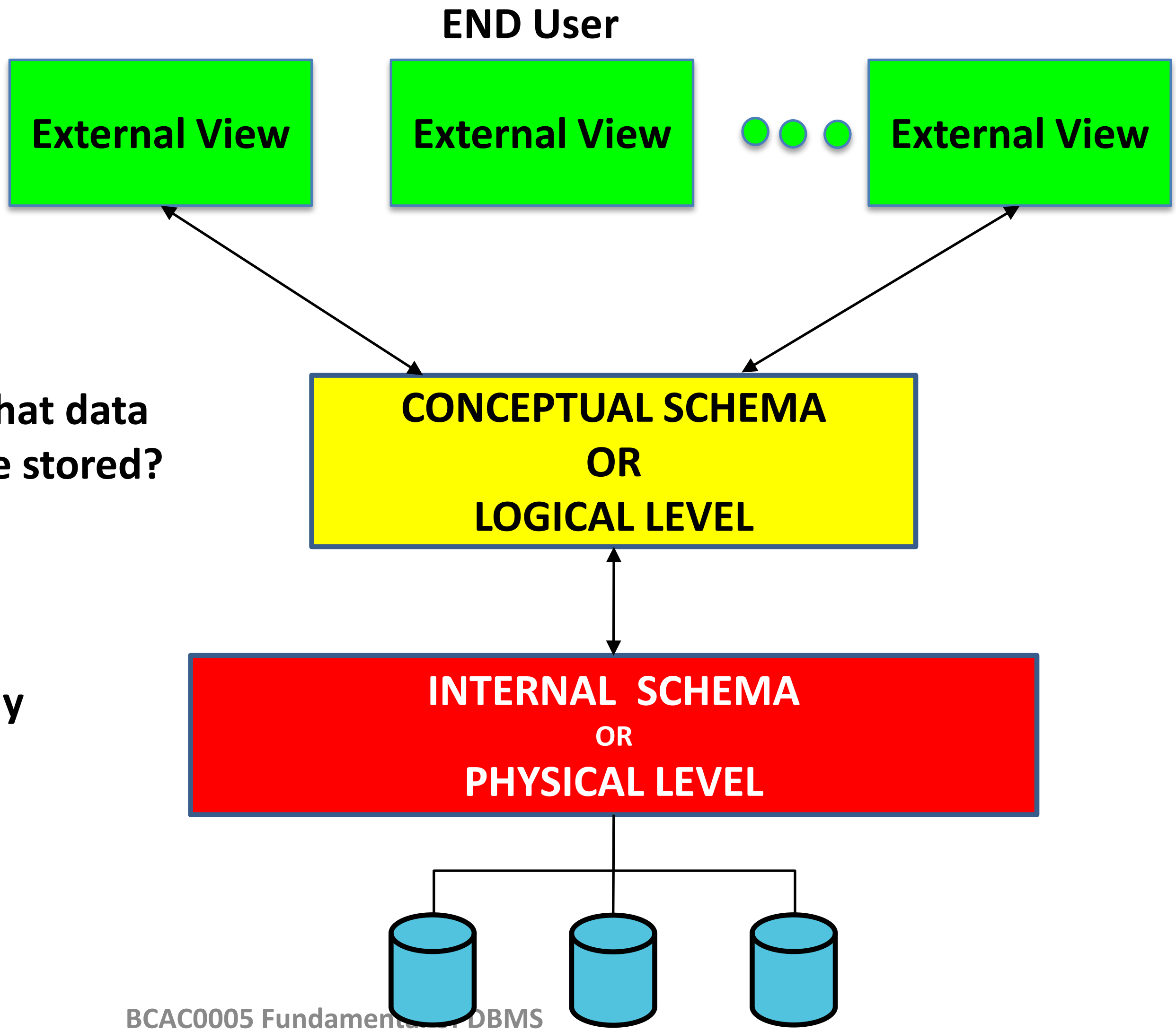
- **When no DBMS may suffice:**
  - If the database system is not able to handle the complexity of data because of modeling limitations
  - If the database users need special operations not supported by the DBMS.

Three Schema Architecture

Forms, Login Pages, Command prompt

What data are stored?

How data actually stored?



END User

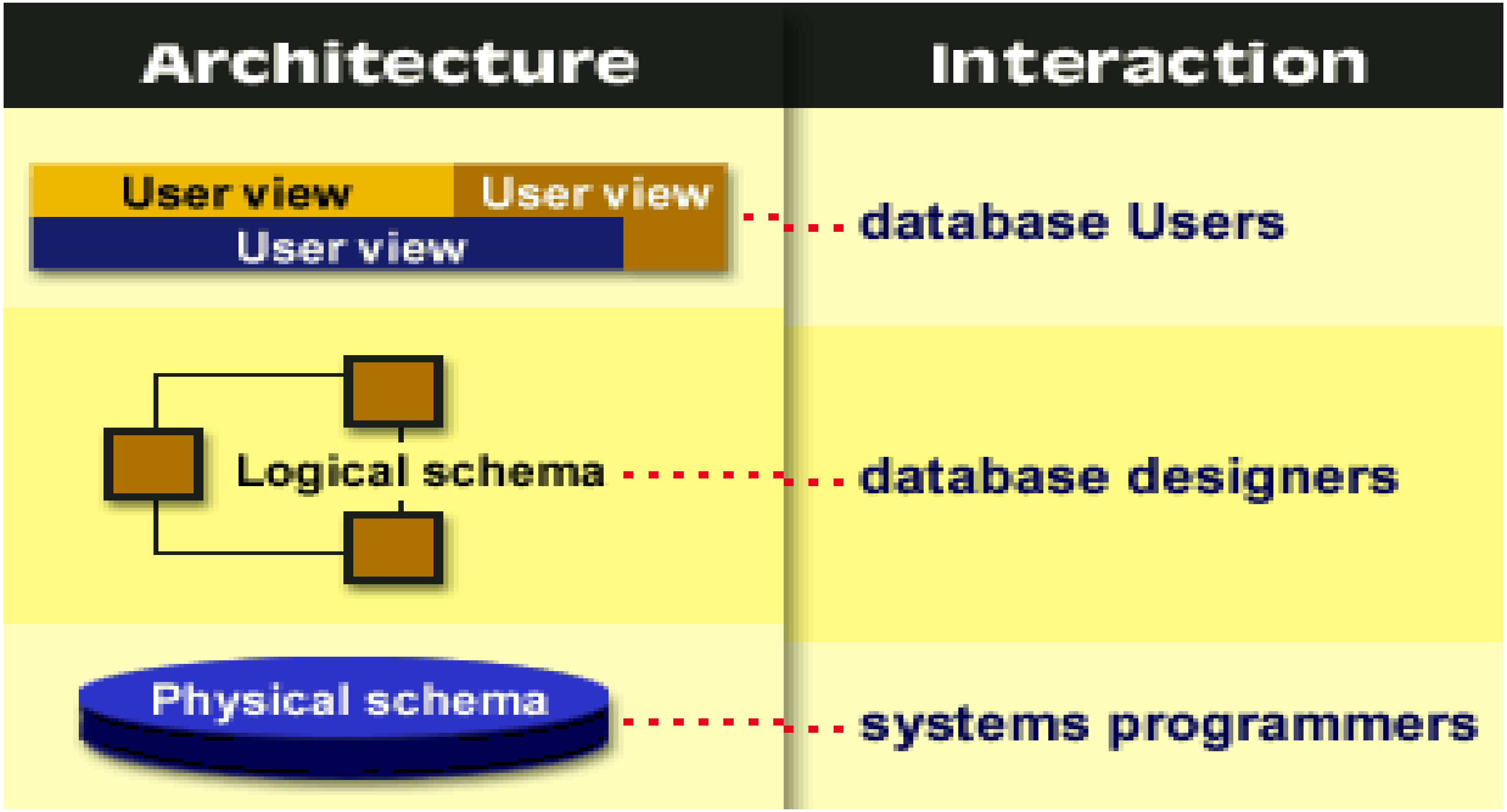
External View

External View

External View

CONCEPTUAL SCHEMA
OR
LOGICAL LEVEL

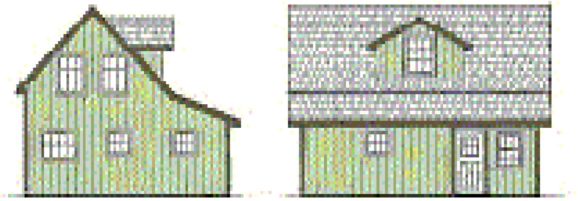
INTERNAL SCHEMA
OR
PHYSICAL LEVEL


Architecture
Interaction
User view
User view
User view
database Users
Logical schema
database designers
Physical schema
systems programmers

Instances and Schemas

- **Logical Schema** – the overall logical structure of the database
  - Example: The database consists of information about a set of customers and accounts in a bank and the relationship between them
    - Analogous to type information of a variable in a program
- **Physical schema**– the overall physical structure of the database
- **Instance** – the actual content of the database at a particular point in time
    - Analogous to the value of a variable

Instances and Schemas

<table>
<tr><th></th><th>Real World</th><th>Database</th></tr>
<tr><td>Scheme</td><td>
Plan for a Standard-House</td><td><table><tr><th>ROLL NO</th><th>NAME</th><th>CGPA</th></tr><tr><td></td><td></td><td></td></tr><tr><td></td><td></td><td></td></tr></table>Template for a Table</td></tr>
<tr><td>Instances</td><td>
Built Standard-Houses</td><td><table><tr><th>ROLL NO</th><th>NAME</th><th>CGPA</th></tr><tr><td>21</td><td>SAN</td><td>8.4</td></tr><tr><td>25</td><td>DAN</td><td>9.1</td></tr></table>Data-filled Tables</td></tr>
</table>

https://selfstudynote.blogspot.com/2016/05/database-schema-and-instance.html

Logical Data Independence

- Logical data is data about database, that is, it stores information about how data is managed inside.
- For example, a table (relation) stored in the database and all its constraints, applied on that relation.
- Logical data independence is a kind of mechanism, which liberalizes itself from actual data stored on the disk.
- If we do some changes on table format, it should not change the data residing on the disk.

Logical Data Independence

- Example: if we add some new columns or remove some columns from table then the user view and programs should not change.
- For example: consider two users A & B.
- Both are selecting the fields "EmployeeNumber" and "EmployeeName".
- If user B adds a new column (e.g. salary) to his table, it will not effect the external view for user A, though the internal schema of the database has been changed for both users A & B.

Logical Data Independence

- **Logical data independence** is more **difficult to achieve** than physical data independence,
- Since **application programs are heavily dependent on the logical structure of the data** that they access.

Physical Data Independence

- The ability to modify the physical schema without changing the logical schema
  - Applications depend on the logical schema
  - In general, the interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.

DBMS Architecture

- A database architect develops and implements software to meet the needs of users.
- The design of a DBMS depends on its architecture.
- It can be centralized or decentralized or hierarchical.
- The architecture of a DBMS can classified into:

1-tier architecture

2-tier architecture

3-tier architecture

Source: https://medium.com/oceanize-geeks/concepts-of-database-architecture-dfdc558a93e4

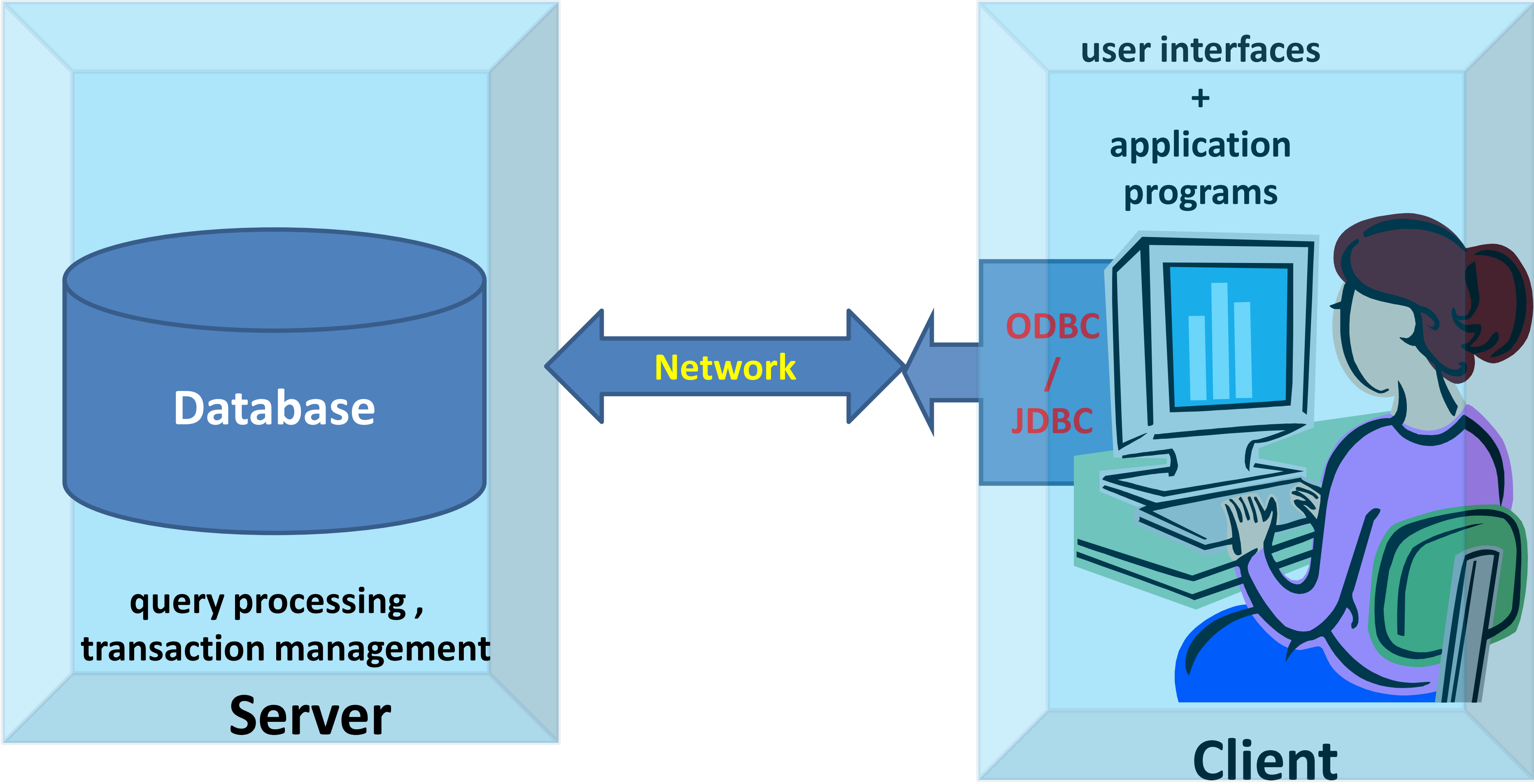
1-tier architecture

- The database is directly available to the user.
- Any changes done here will directly be done on the database itself.
- It doesn't provide a handy tool for end users.
- The 1-Tier architecture is used for development of the local application, where programmers can directly communicate with the database for the quick response.

2-tier architecture

- Same as basic client-server.
- **Client side** (The user interfaces and application programs)
- **Server side** (Query processing and transaction management over database)
- Applications on the client end can directly communicate with the database at the server side.
- To communicate with the DBMS, client-side application establishes a connection with the server side.
- For this interaction, API's like: **ODBC**, **JDBC** are used.

2-tier architecture



Database

query processing ,
transaction management

Server

Network

ODBC
/
JDBC

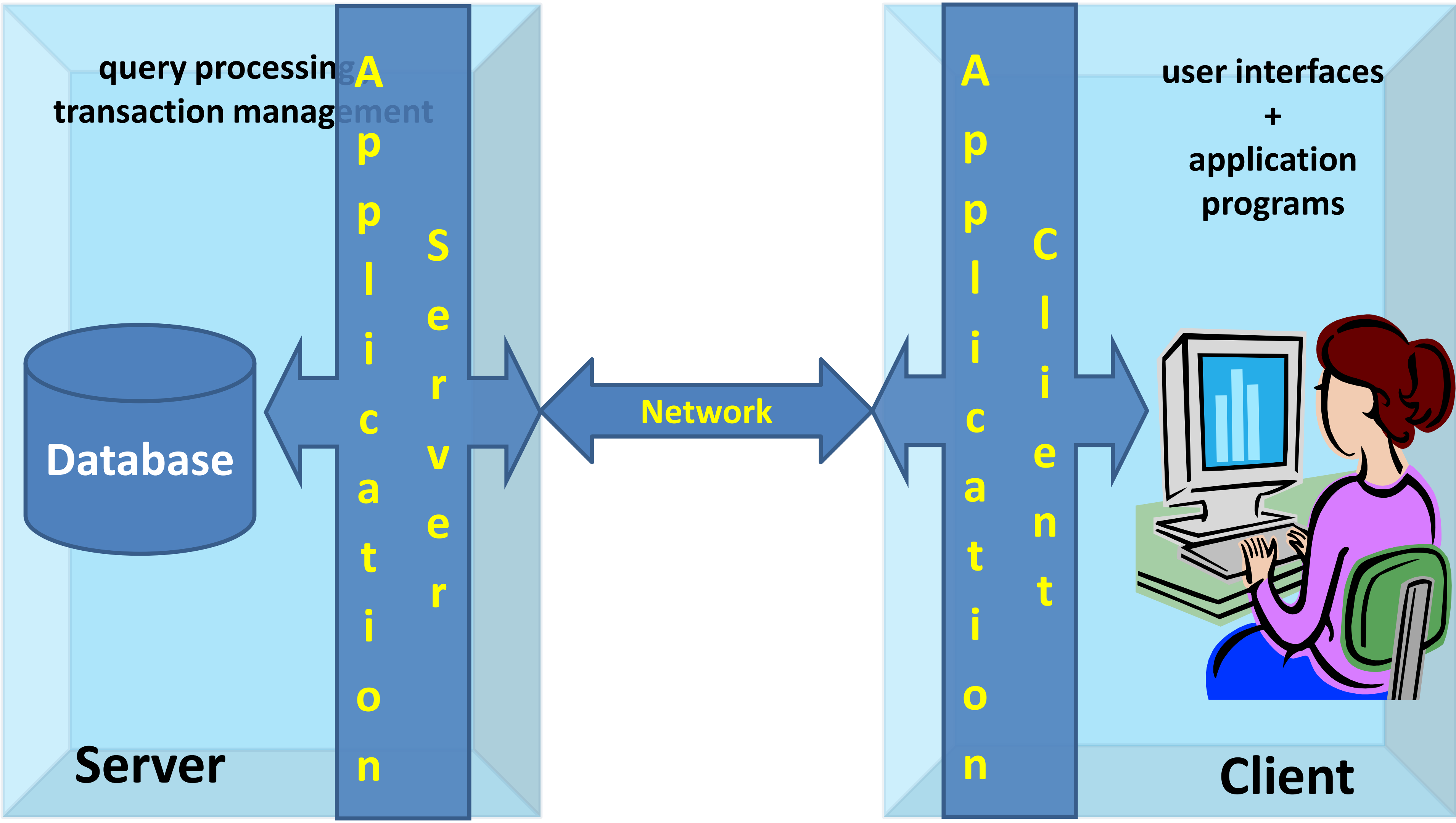
user interfaces
+
application
programs

Client

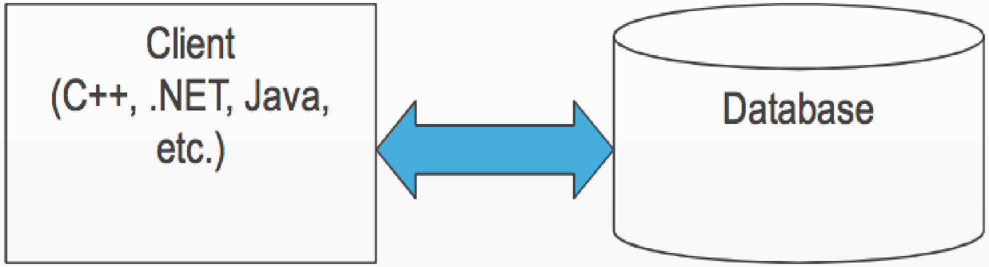
2-tier architecture

- Advantage
  - Maintenance and understanding is easier
- Disadvantage
  - Poor performance when there are a large number of users.

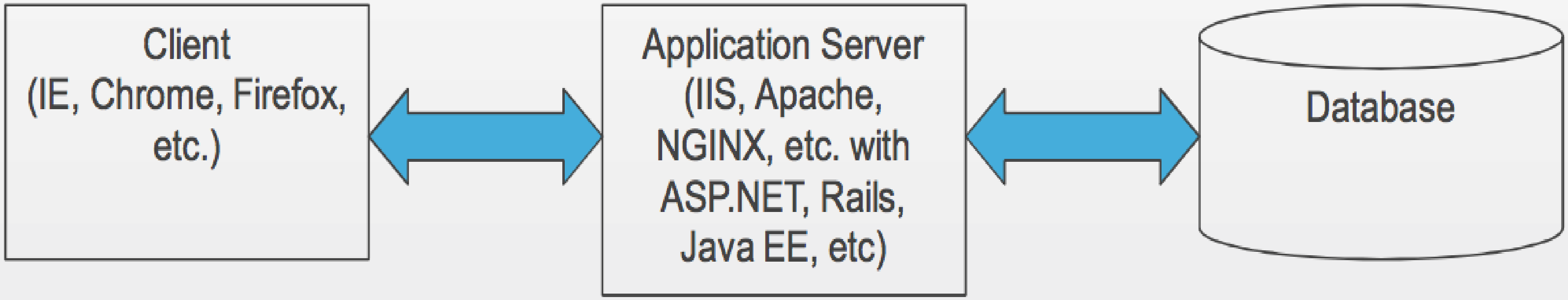
3-tier architecture



2-Tiered Architecture



3-Tiered Architecture


Client
(IE, Chrome, Firefox, etc.)
Application Server
(IIS, Apache, NGINX, etc. with ASP.NET, Rails, Java EE, etc)
Database

3-tier architecture

- Application layer (business logic layer) between the user and the DBMS
- Application layer is responsible for communicating the user's request to the DBMS system and send the response from the DBMS to the user.
- Application layer processes
  - functional logic, constraint, and rules

    before passing data to the user or down to the DBMS
- Three tier architecture is the most popular DBMS architecture.

3-tier architecture

The goal of Three-tier architecture is:

- To separate the user applications and physical database
- Proposed to support DBMS characteristics
- Program-data independence
- Support of multiple views of the data
- This type of architecture is used in case of large web applications.

Question ?

Where we use 1-tier/2-tier/3-tier Architecture ?

Answer

1-tier:

Development of the local application

2-tier:

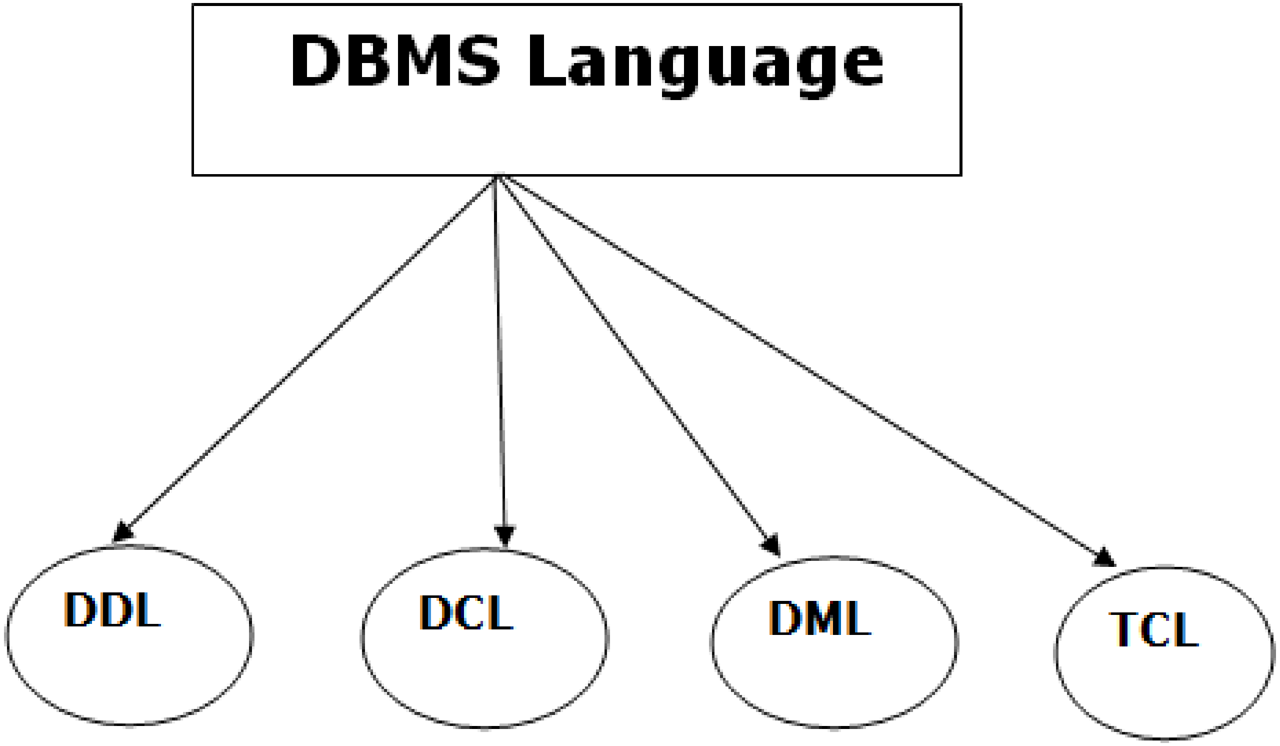
Attendance Management System

Library Management System

3-tier: Any large website

IRCTC, facebook, etc.

Database Language



1. Data Definition Language

- Define **database structure**.
- It is used to create,
  - Schema, tables, indexes, constraints, etc.

  in the database.
- Used to store the information of **metadata**
- **Metadata**
  - the number of tables and schemas,
  - their names,
  - indexes,
  - columns in each table,
  - constraints, etc.

Tasks that come under DDL

- These commands are used to **update the database schema** that's why they come under Data definition language.
  - **Create:** It is used to create objects in the database.
  - **Alter:** It is used to alter the structure of the database.
  - **Drop:** It is used to delete objects from the database(removes structure).
  - **Truncate:** It is used to remove all records from a table (structure remains same).
  - **Rename:** It is used to rename an object.
  - **Comment:** It is used to comment on the data dictionary.

2. Data Manipulation Language

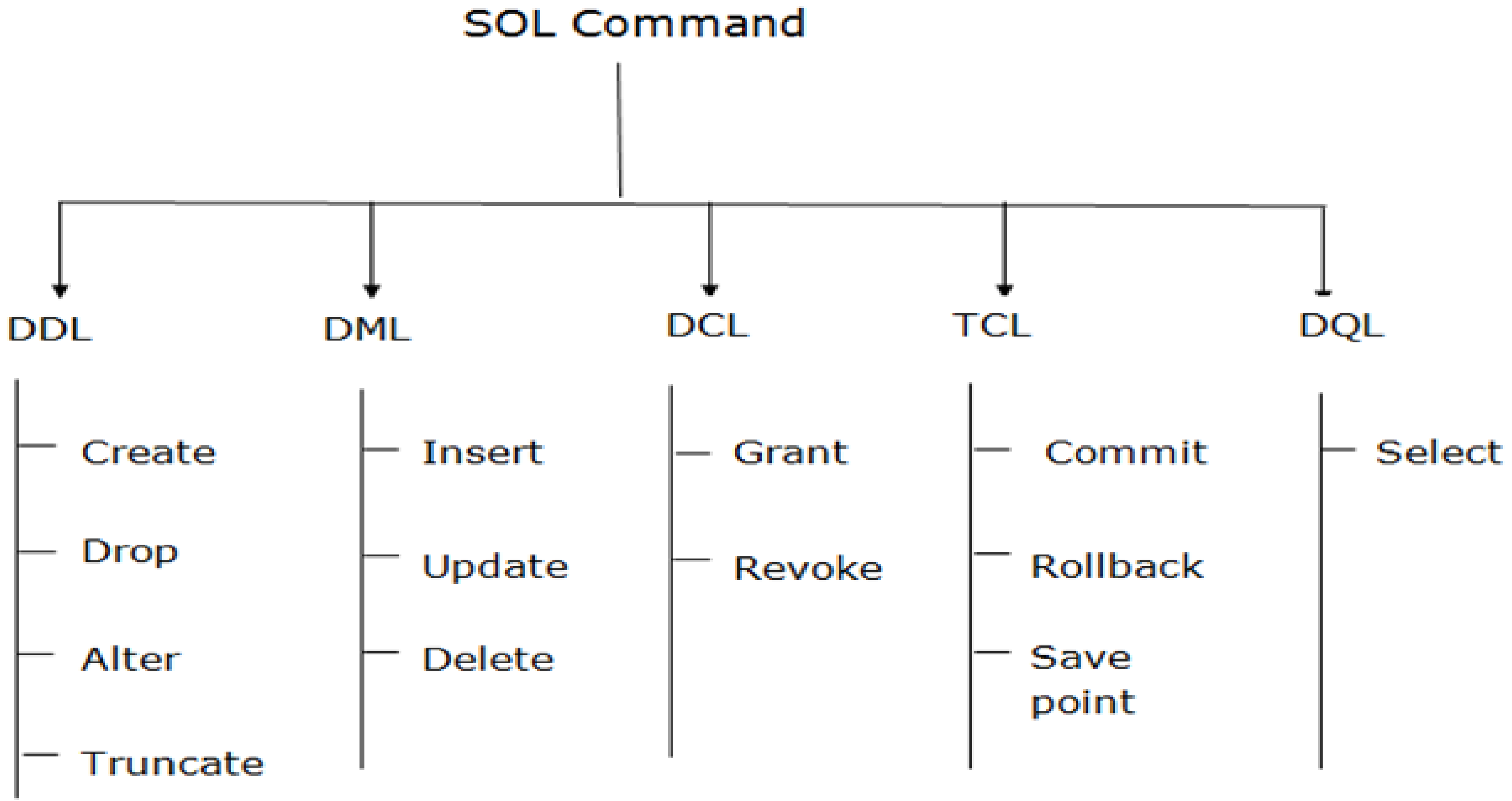
- It is used for accessing and manipulating data in a database.
- It handles user requests.
  - **Select:** It is used to retrieve data from a database.
  - **Insert:** It is used to insert data into a table.
  - **Update:** It is used to update existing data within a table.
  - **Delete:** It is used to delete few(based on condition)/all records from a table(structure remain same).

3. Data Control Language

- The DCL execution is transactional.
- Tasks that come under DCL:
  - **Grant:** It is used to give user access privileges to a database.
  - **Revoke:** It is used to take back permissions from the user.

4. Transaction Control Language

- TCL is used to run the changes made by the DML statement.
- TCL can be grouped into a logical transaction.
- Tasks that come under TCL:
  - **Commit:** It is used to save the transaction on the database.
  - **Rollback:** It is used to restore the database to original since the last Commit.



SOL Command

- DDL
  - Create
  - Drop
  - Alter
  - Truncate
- DML
  - Insert
  - Update
  - Delete
- DCL
  - Grant
  - Revoke
- TCL
  - Commit
  - Rollback
  - Save point
- DQL
  - Select

Question ?

- **Drop** vs **Delete** vs **Truncate……..?**

Drop vs **Delete** vs **Truncate**

Delete	Truncate	Drop
DML	DDL	DDL
Removes few or all records	Removes all record	Removes table
No change in structure	No Change in structure	Change in structure
May use **where** clause	Not used	Not used
Can be rollback	Can’t rollback	Can’t rollback
Slower than truncate	Faster than Delete	Faster than Delete

Database model

- It shows the **logical structure of a database**, including the relationships and constraints
- It determine how data can be stored and accessed.
- Most data models can be represented by an accompanying **database diagram**.

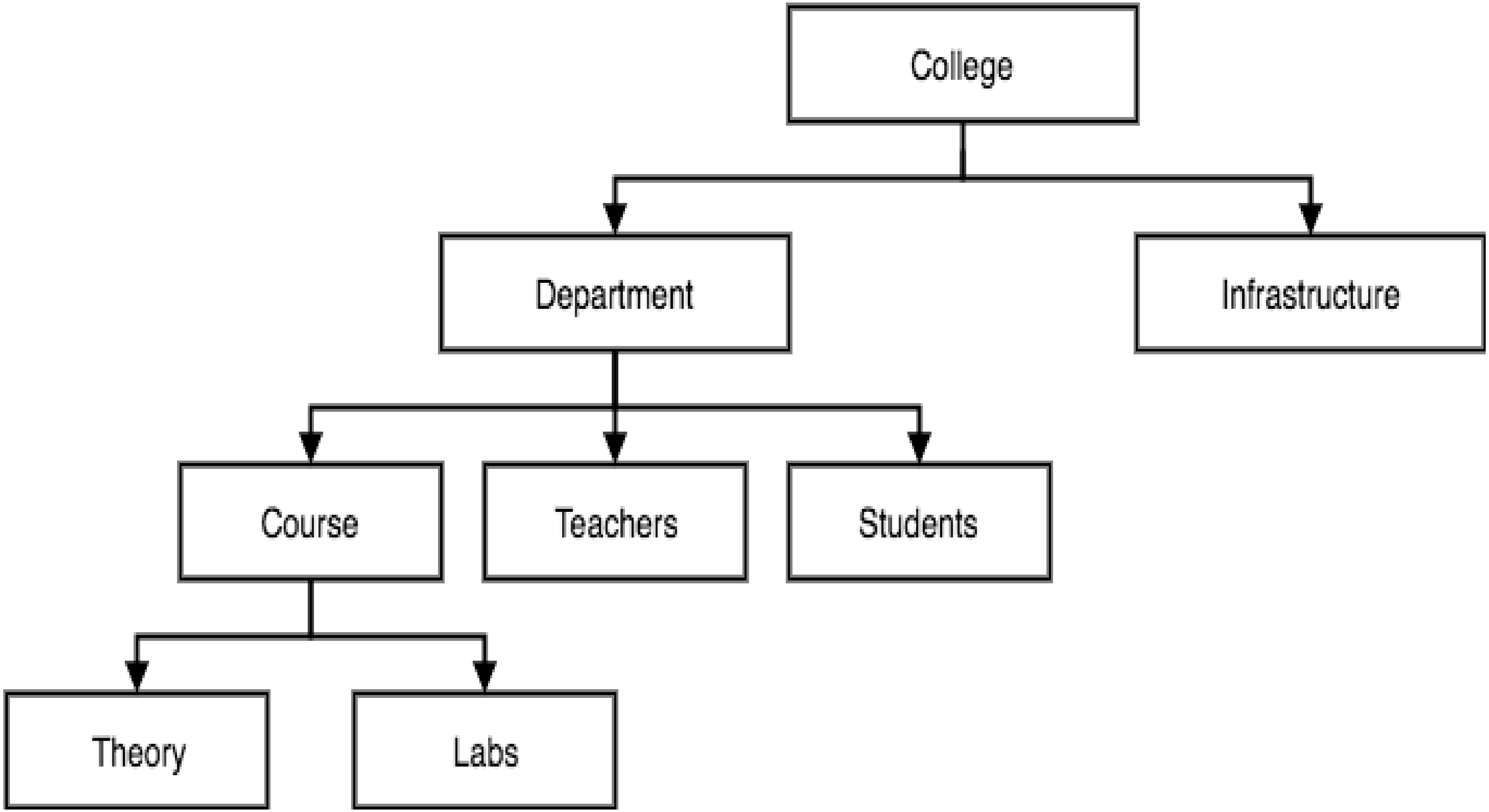
Types of database models

- Hierarchical database model
- Network model
- Relational model
- Object-oriented database model

Hierarchical model

- The hierarchical model **organizes data into a tree-like structure**.
- Each record has a single parent or root.
- Sibling records are sorted in a particular order.
- That order is used as the physical order for storing the database.
- This model is good for describing many real-world relationships.
- This model was primarily used by IBM’s Information Management Systems in the 60s and 70s, but they are rarely seen today due to certain operational inefficiencies.

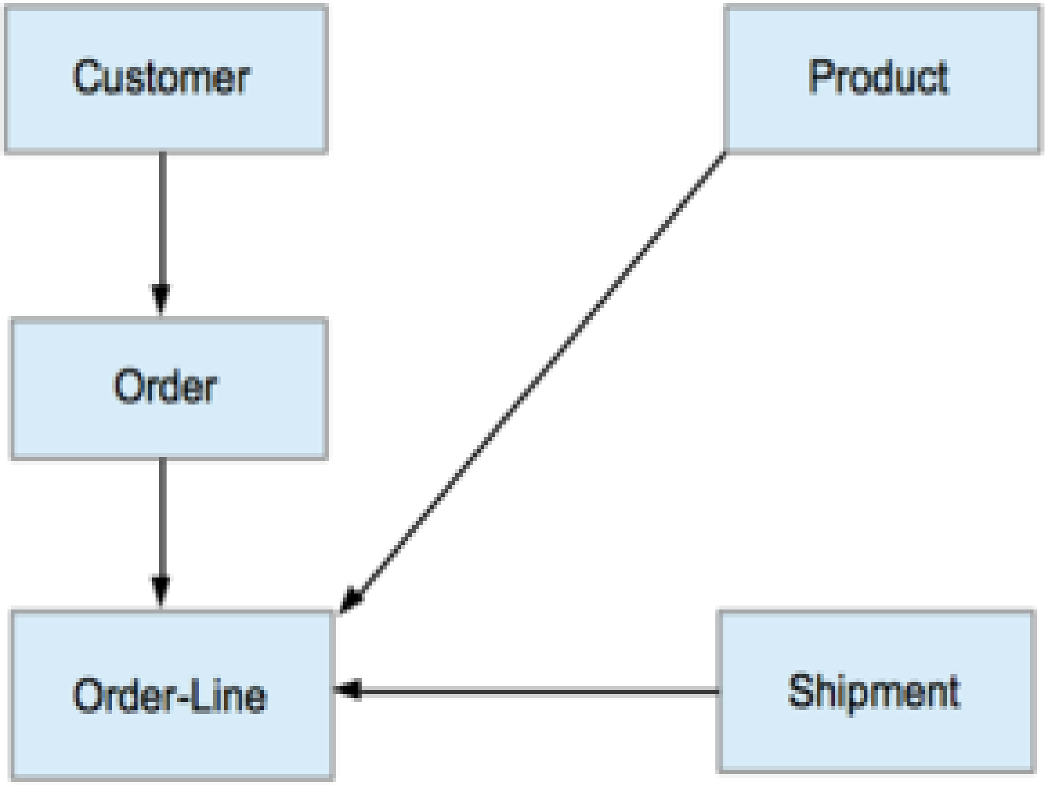
Hierarchical model



Network Model

- This is an extension of the Hierarchical model.
- In this model data is organized more like a graph, and are allowed to have more than one parent node.
- In this database model data is more related as more relationships are established in this database model.
- Also, as the data is more related, hence accessing the data is also easier and fast.
- This database model was used to map many-to-many data relationships.
- This was the most widely used database model, before Relational Model was introduced.

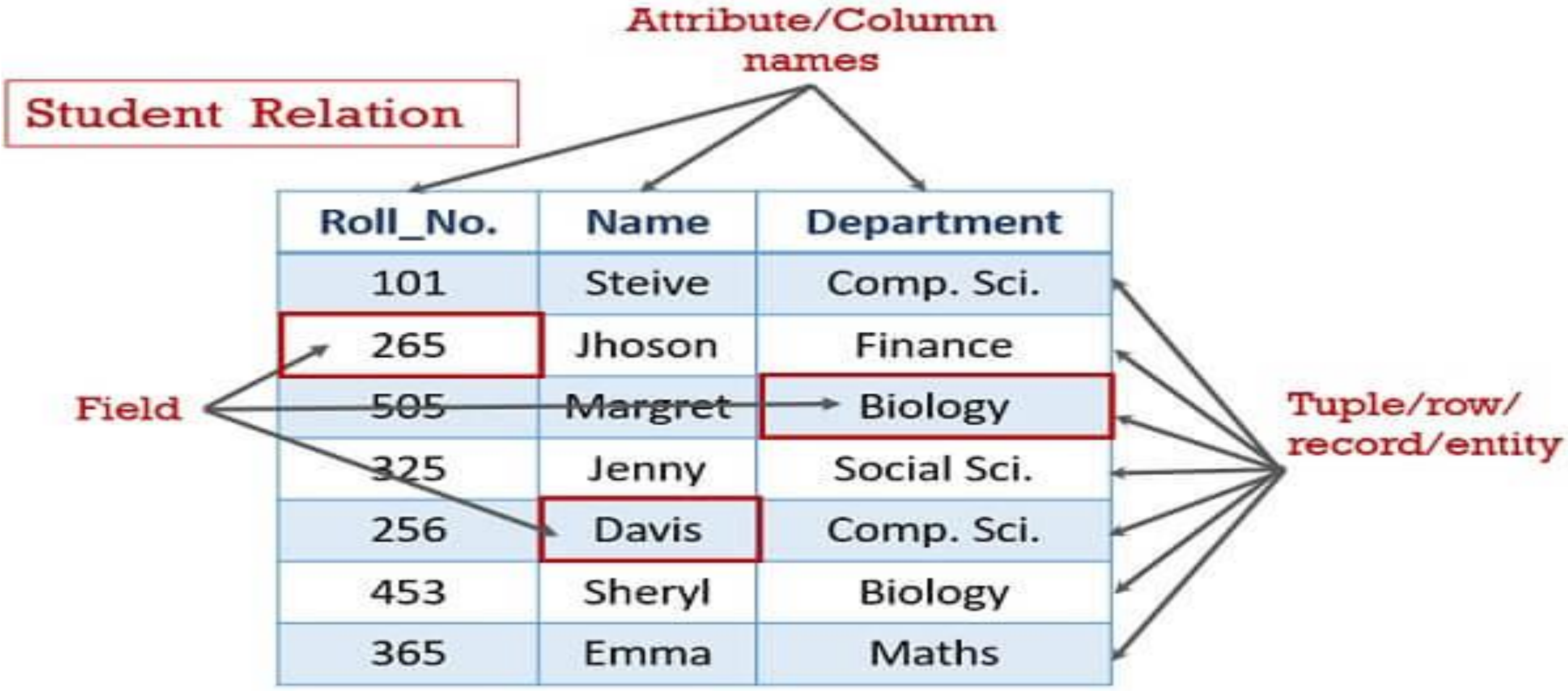
Network Model



Relational model

- The model was introduced by E.F. Codd in 1970.
- The most common model, the relational model store data into tables,
- Tables are also known as relations, each of which consists of columns and rows.
- Each column lists an attribute of the entity, such as price, zip code, or birth date.
- Together, the attributes in a relation are called a domain.
- A particular attribute or combination of attributes is chosen as a primary key that can be referred to in other tables, it’s called a foreign key.

- Each row, also called a tuple, includes data about a specific instance of the entity, such as a particular employee.
- The model also accounts for the types of relationships between those tables, including one-to-one, one-to-many, and many-to-many relationships.
- Relational databases are typically written in Structured Query Language (SQL).



Student Relation

Attribute/Column names

Roll_No.	Name	Department
101	Steive	Comp. Sci.
265	Jhoson	Finance
505	Margret	Biology
325	Jenny	Social Sci.
256	Davis	Comp. Sci.
453	Sheryl	Biology
365	Emma	Maths

Field

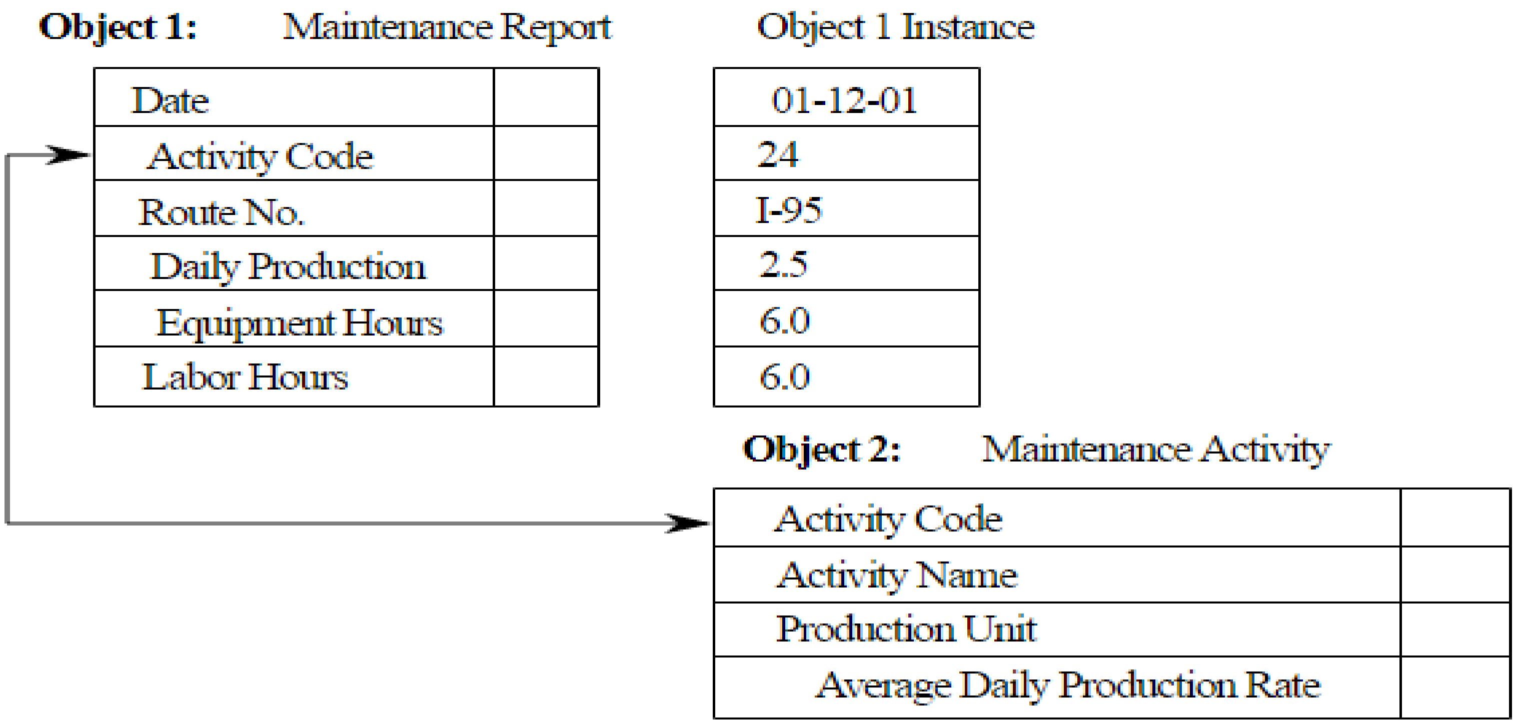
Tuple/row/record/entity

Student Relation in Relational Model

Object Oriented Data model

- An **object database** is a database management system in which information is represented in the form of objects as used in object-oriented programming.
- Object databases are different from relational databases which are table-oriented.
- Object databases have been considered since the early 1980s

Object-Oriented Model



Object 1: Maintenance Report

Date	
Activity Code	
Route No.	
Daily Production	
Equipment Hours	
Labor Hours	

Object 1 Instance

01-12-01
24
I-95
2.5
6.0
6.0

Object 2: Maintenance Activity

Activity Code	
Activity Name	
Production Unit	
Average Daily Production Rate	

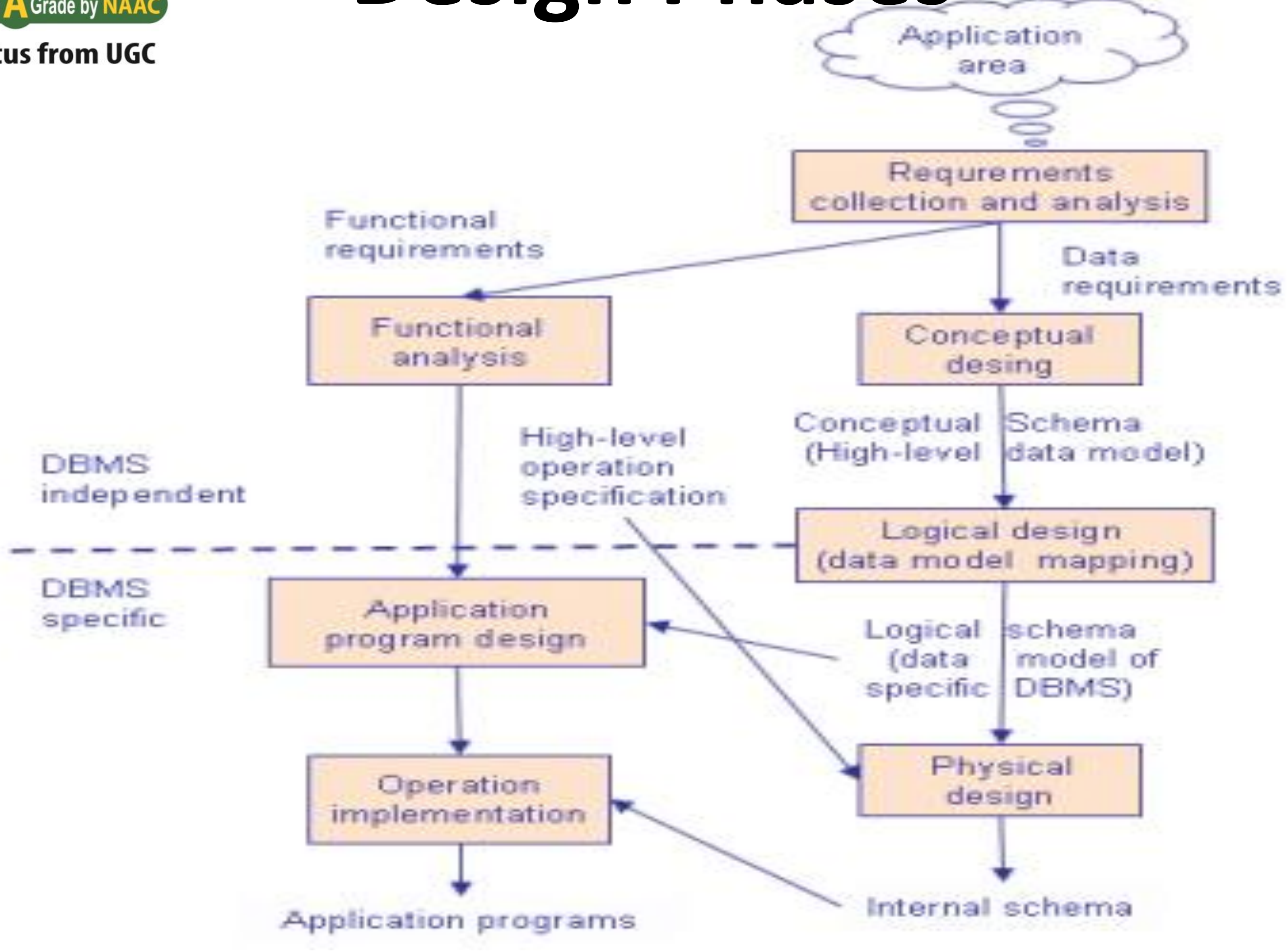
Questions ?

- **Which database model is used in your University ?**

Answer

- **Relational Model**

Design Phases


Application area
Requrements collection and analysis
Functional requirements
Data requirements
Functional analysis
Conceptual desing
High-level operation specification
Conceptual Schema (High-level data model)
DBMS independent
Logical design (data model mapping)
DBMS specific
Application program design
Logical schema (data model of specific DBMS)
Operation implementation
Physical design
Application programs
Internal schema

Design Phases

1. **Conceptual Design**
   Once all the requirements have been collected and analyzed, the next step is to create a conceptual schema for the database, using a high level conceptual data model. The **result** of this phase is an **Entity-Relationship (ER) diagram or UML class diagram**. It describes how different entities (objects, items) are related to each other. It also describes what attributes (features) each entity has. It includes the definitions of all the concepts (entities, attributes) of the application area. During or after the conceptual schema design, the basic data model operations can be used to specify the high-level user operations identified during the functional analysis. This also serves to confirm that the conceptual schema meets all the indentified functional requirements.

Input: Data Requirements
Output: ER Diagram

Design Phases

2. **Logical Design**
   The **result** of the logical design phase (or data model mapping phase) is a set of **relation schemas**. The ER diagram or class diagram is the basis for these relation schemas. To create the relation schemas is quite a mechanical operation. **There are rules how the ER model or class diagram is transferred to relation schemas.** The relation schemas are the basis for table definitions. In this phase (if not done in previous phase) the **primary keys and foreign keys are defined**.

Input: ER Diagram
Output: Relation Schemas

Design Phases

Normalization

Normalization is the last part of the logical design. The goal of normalization is to eliminate redundancy and potential update anomalies. Redundancy means that the same data is saved more than once in a database. Update anomaly is a consequence of redundancy. If a piece of data is saved in more than one place, the same data must be updated in more than one place. Normalization is a technique by which one can modify the relation schema to reduce the redundancy. Each normalization phase adds more relations (tables) into the database.

Input: Relation Schemas

Output: Refined Relation Schemas

Design Phases

Physical Design

The goal of the last phase of database design, physical design, is to **implement the database**. At this phase one must know **which database management system (DBMS) is used**. For example, different DBMS's have different names for datatypes and have different datatypes. The SQL clauses to create the database are written. The idexes, the integrity constraints (rules) and the users' access rights are defined. Finally the **data to test the database is added in**.

Input: Refined Relation Schemas

Output: Implement the database

Design Phases

- **Initial phase** -- characterize fully the data needs of the prospective database users.
- **Second phase** -- choosing a data model
  - Applying the concepts of the chosen data model
  - Translating these requirements into a conceptual schema of the database.
  - A fully developed conceptual schema indicates the functional requirements of the enterprise.
    - Describe the kinds of operations (or transactions) that will be performed on the data.

Design Phases (Cont.)

- **Final Phase** -- Moving from an abstract data model to the implementation of the database
  - **Logical Design** – Deciding on the database schema.
    - Database design requires that we find a “good” collection of relation schemas.
    - Business decision – What attributes should we record in the database?
    - Computer Science decision – What relation schemas should we have and how should the attributes be distributed among the various relation schemas?
  - **Physical Design** – Deciding on the physical layout of the database

Design Approaches

- **Entity Relationship Model**
  - Models an enterprise as a collection of *entities* and *relationships*
    - Entity: a “thing” or “object” in the enterprise that is distinguishable from other objects
      - Described by a set of *attributes*
    - Relationship: an association among several entities
  - Represented diagrammatically by an *entity-relationship diagram:*
- **Normalization Theory**
  - Formalize what designs are bad, and test for them

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-
  - Entity Sets
  - Attributes
  - Relationship Set

Example-

- This complete table is referred to as “Student Entity Set” and

- Each row represents an “entity”.

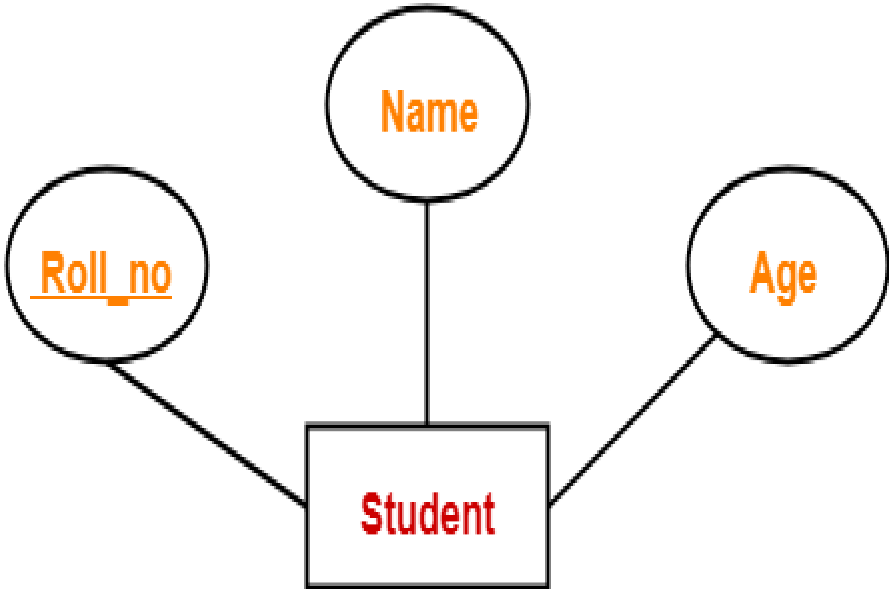
Student Table

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

Representation as ER Diagram

Student Table

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



ER Diagram Symbols- Entity Sets

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-
  - Strong entity set
  - Weak entity set

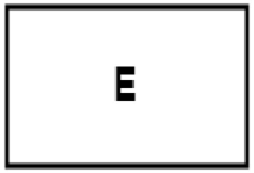
ER Diagram Symbols- Entity Sets

1. Strong Entity Set-

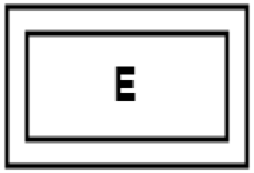
- A strong entity set possess its own primary key.
- It is represented using a single rectangle.

2. Weak Entity Set-

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



Strong Entity Set



Weak Entity Set

ER Diagram Symbols-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-
  - Strong relationship set
  - Weak relationship set

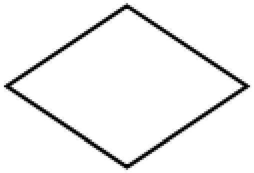
ER Diagram Symbols-Relationship Sets

1. **Strong Relationship Set-**

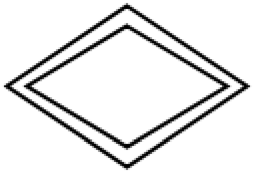
- A strong relationship exists between two strong entity sets.
- It is represented using a diamond symbol.

2. Weak Relationship Set-

- A weak relationship exists between the strong and weak entity set.
- It is represented using a double diamond symbol.



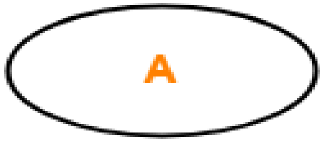
Strong Relationship Set



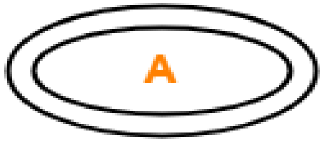
Weak or Identifying Relationship Set

ER Diagram Symbols - Attributes

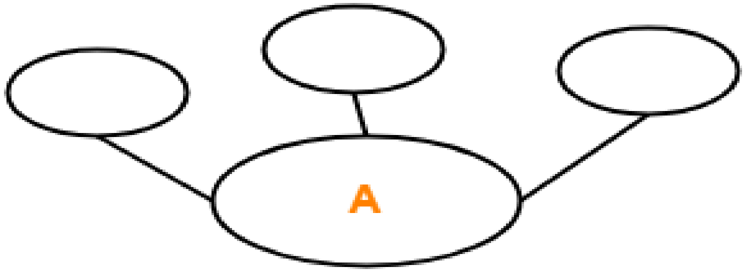
- Attributes are the properties which describes the entities of an entity set.
- There are several types of attributes.



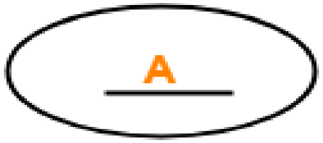
Attribute



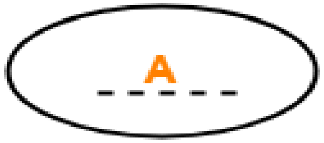
Multivalued Attribute



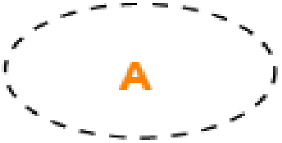
Composite Attribute



Key Attribute



Partial Attribute



Derived Attribute

ER Diagram Symbols- 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-
  - Partial participation
  - Total participation

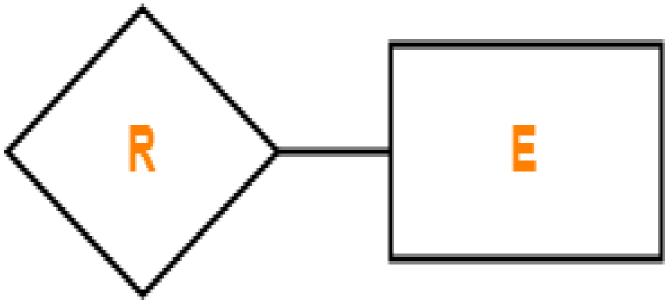
ER Diagram Symbols- Participation Constraints

1. **Partial Participation-**

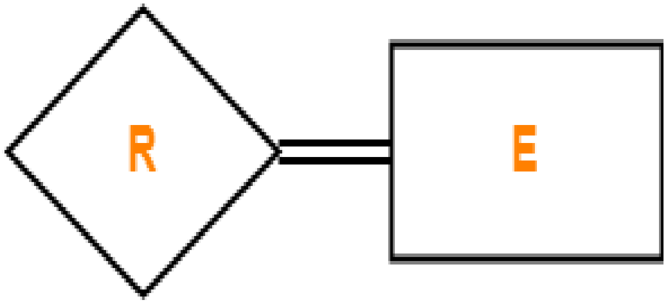
- Partial participation is represented using a single line between the entity set and relationship set.

2. **Total Participation-**

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



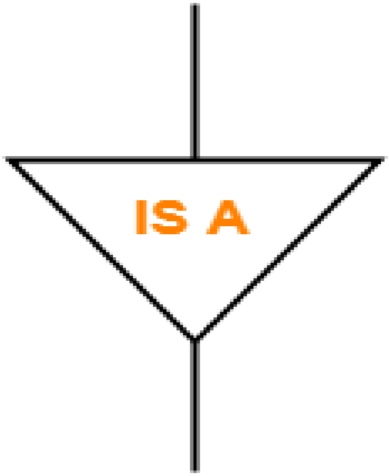
Partial Participation



Total Participation

ER Diagram Symbols – Specialization and Generalization

- **Generalization** is a process of forming a generalized super class by extracting the common characteristics from two or more classes.
- **Specialization** is a reverse process of generalization where a super class is divided into sub classes by assigning the specific characteristics of sub classes to them.



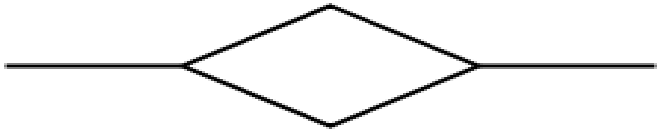
IS A specialization or generalization

ER Diagram Symbols – Cardinality Constraints / Ratios

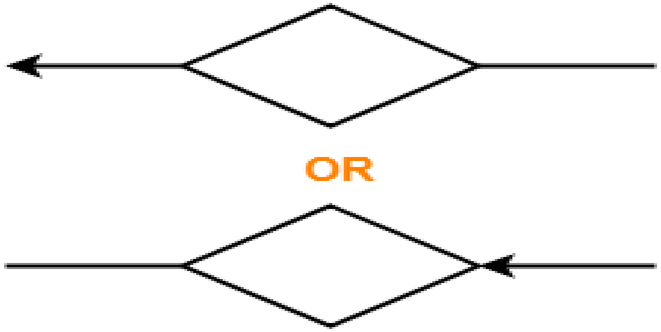
- Cardinality constraint defines the maximum number of relationship instances in which an entity can participate.
- There are 4 types of cardinality ratios-
  - Many-to-many cardinality (m:n)
  - Many-to-one cardinality (m:1)
  - One-to-many cardinality (1:n)
  - One-to-one cardinality (1:1)

ER Diagram Symbols –

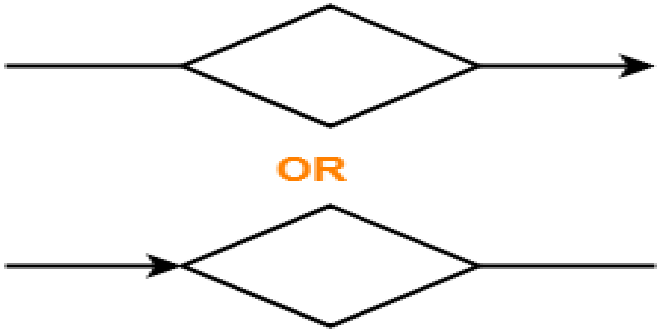
Cardinality Constraints / Ratios



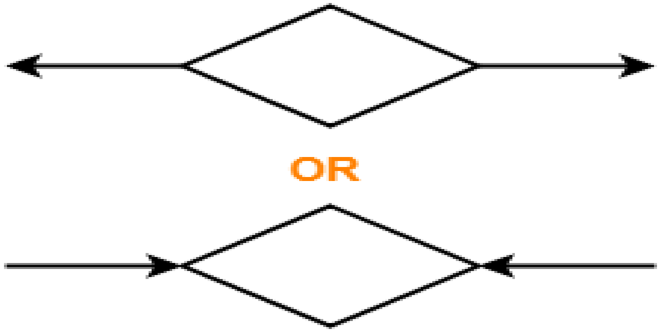
Many-to-Many relationship
(m:n)



One-to-Many relationship
(1:n)



Many-to-One relationship
(m:1)



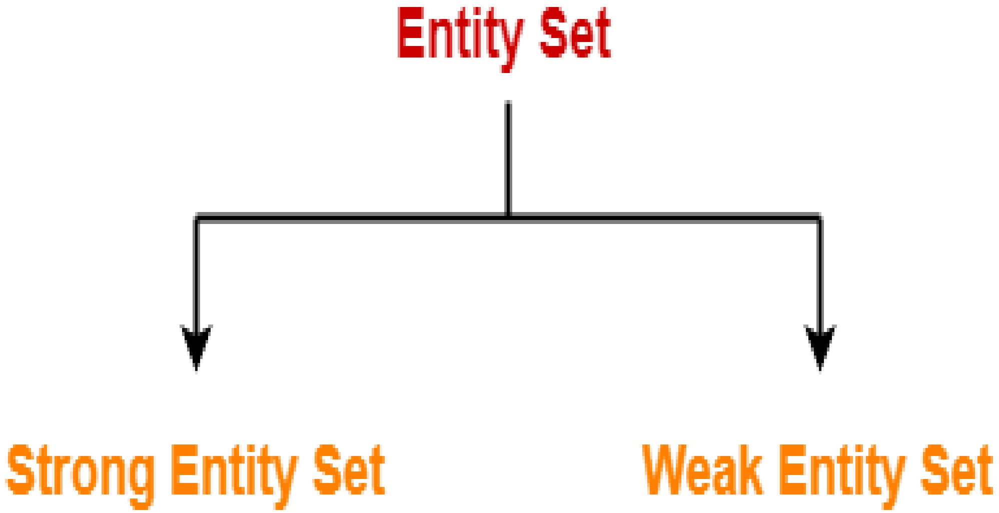
One-to-One relationship
(1:1)

Entity Sets in DBMS

- 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, 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



Entity Set

Strong Entity Set

Weak Entity Set

1. Strong Entity Set

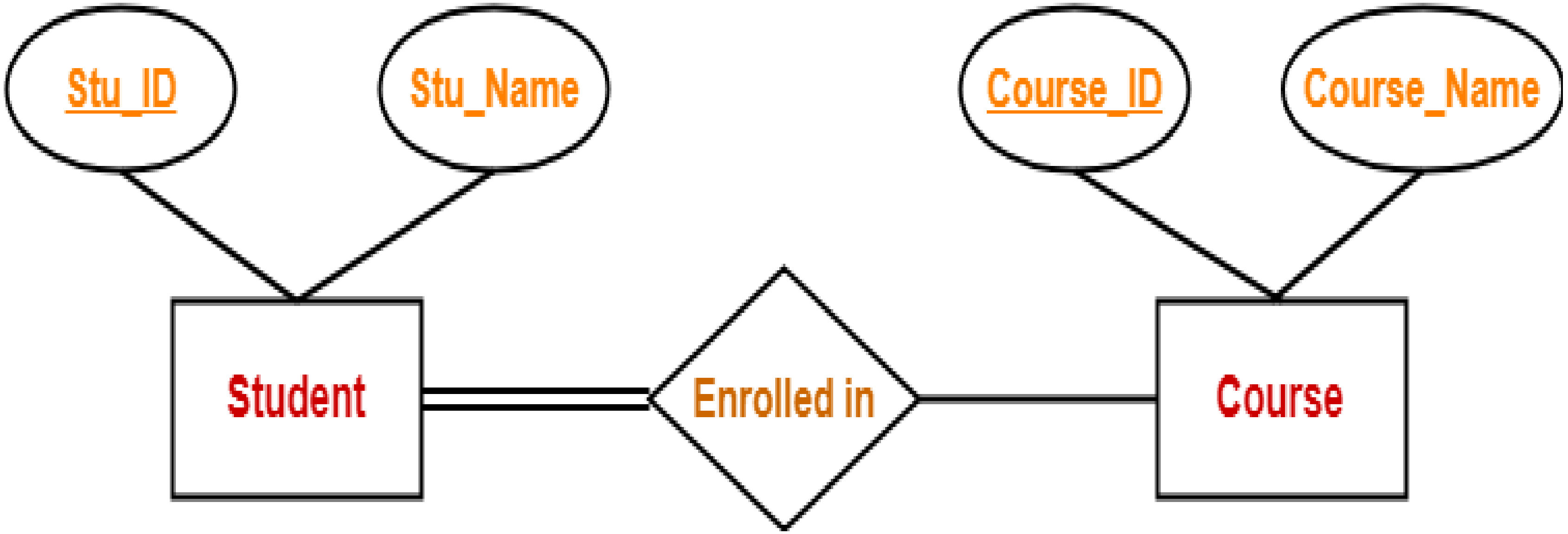
- A strong entity set is an entity set that contains sufficient attributes to uniquely identify all its entities.
- In other words, a primary key exists for a strong entity set.
- Primary key of a strong entity set is represented by underlining it.

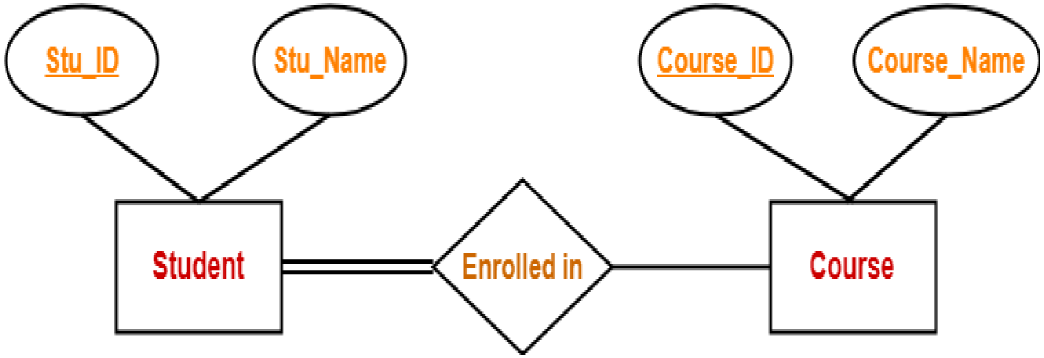
1. Strong Entity Set

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.

1. Strong Entity Set





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.

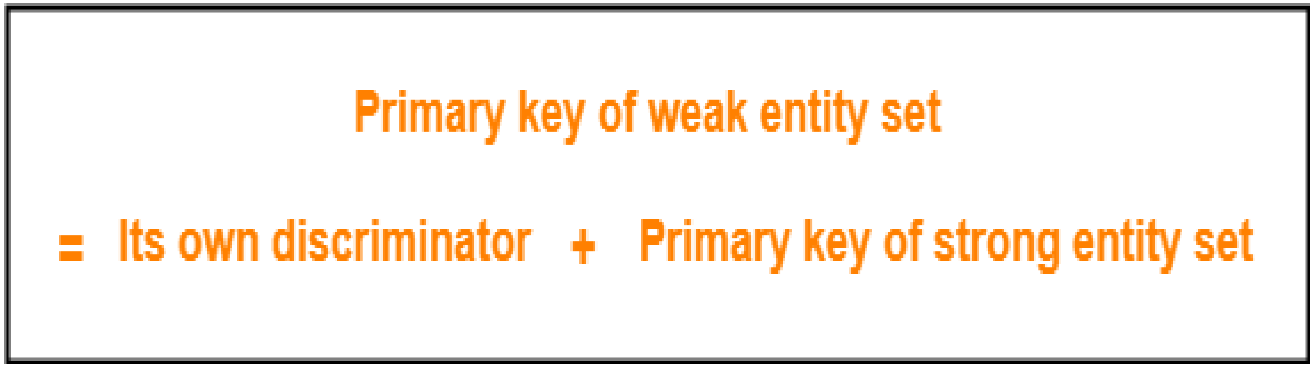
2. Weak Entity Set

- 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

2. Weak Entity Set

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

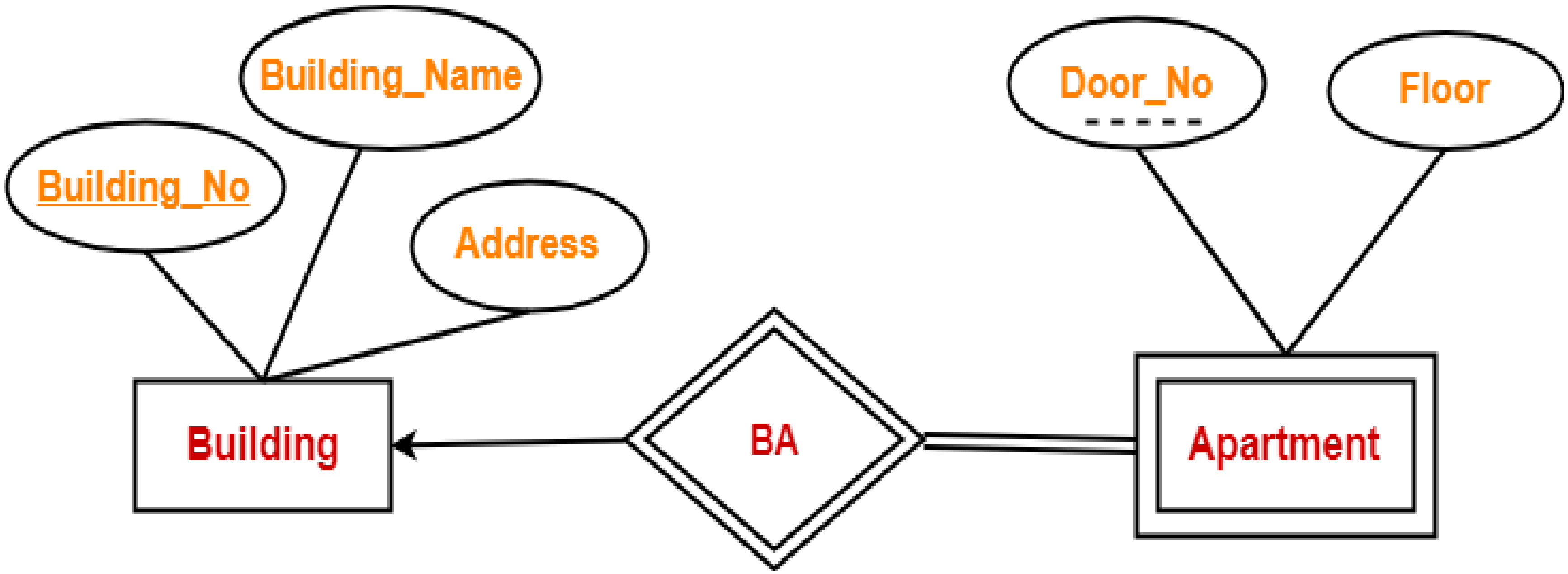
= Its own discriminator + Primary key of strong entity set

2. Weak Entity Set

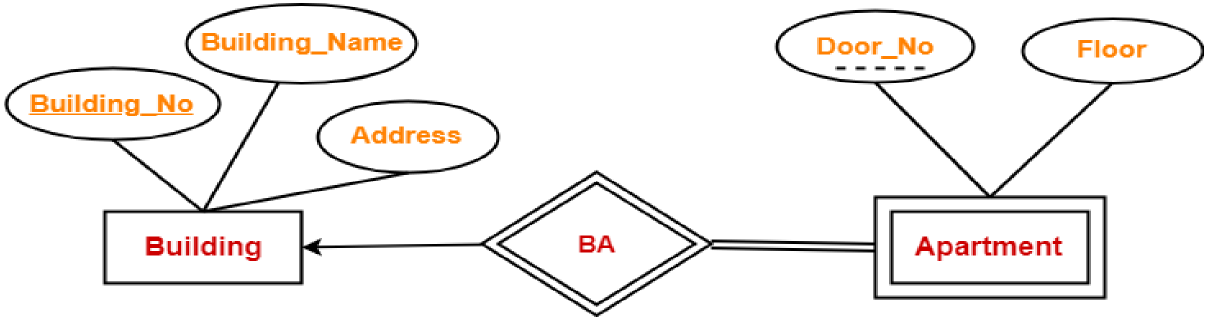
Symbols Used-

- 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.

2. Weak Entity Set



- 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.

2. Weak Entity Set

- 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

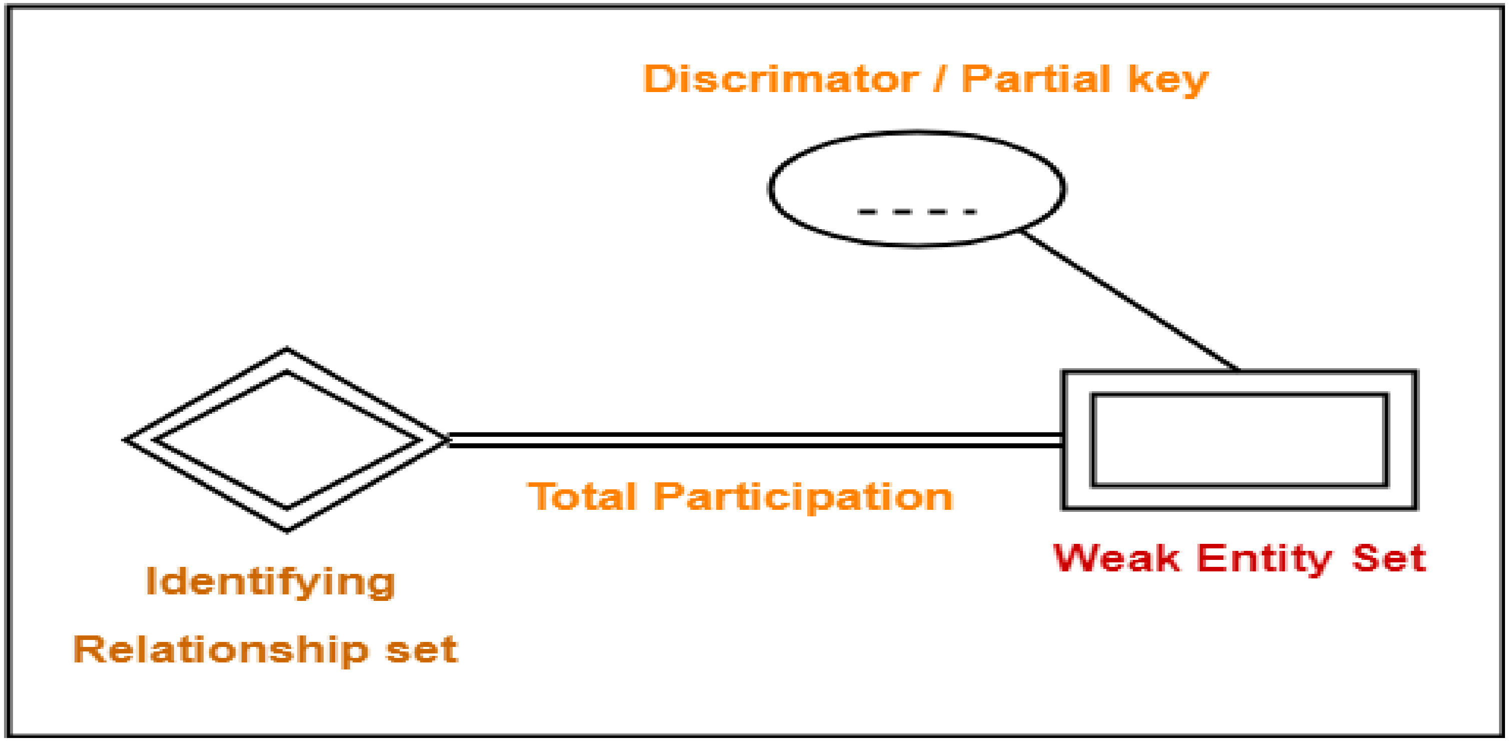
  = Building number + Door number

Strong entity set VS Weak entity set

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-

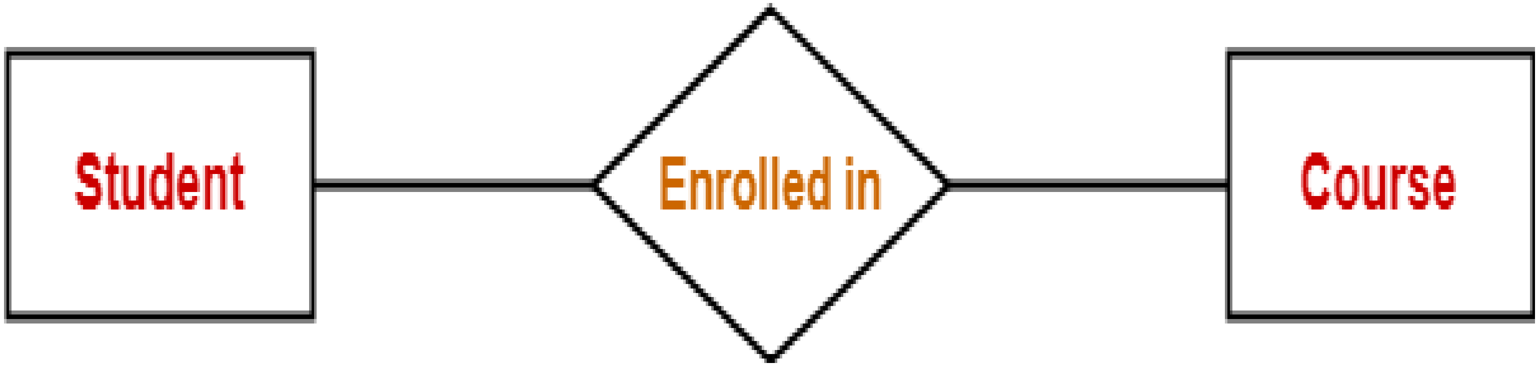


Relationship in DBMS

- A relationship is defined as an association among several entities.

Example

- ‘Enrolled in’ is a relationship that exists between entities **Student** and **Course**.

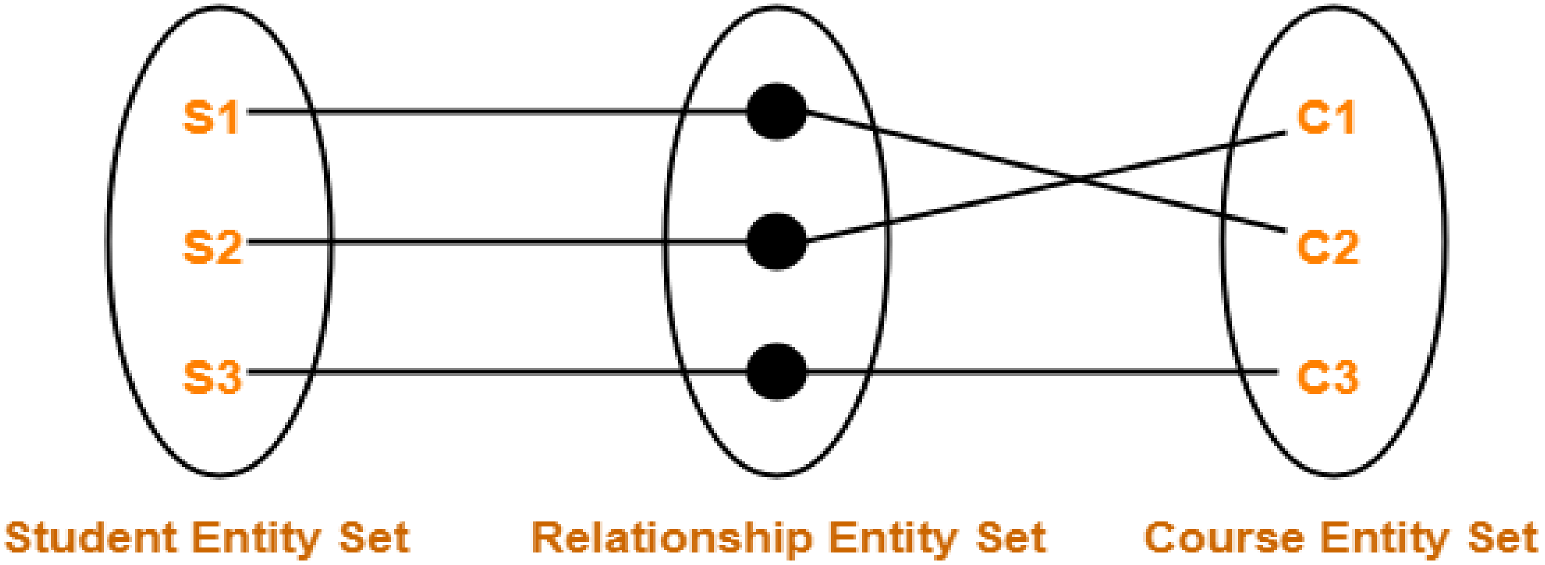


Relationship Set-

- 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

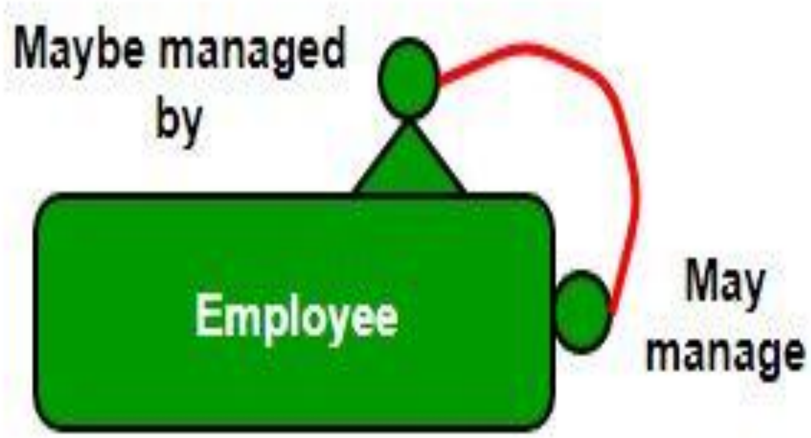
Types of Relationship Sets-

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

1. Unary Relationship Set-

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

Example-

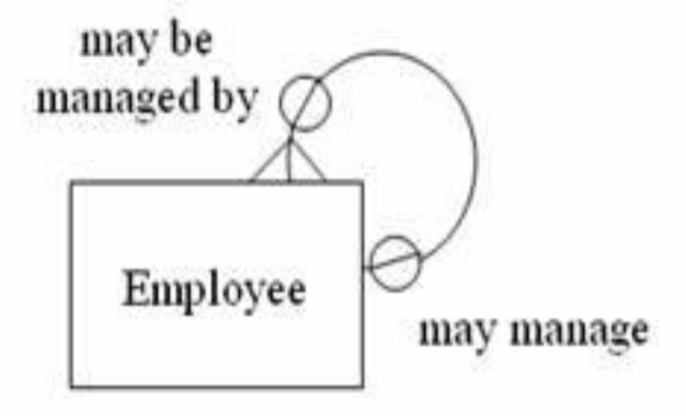


Recursive Relationship

- It is possible for the same entity to participate in the relationship.
- This is termed a **recursive relationship**.

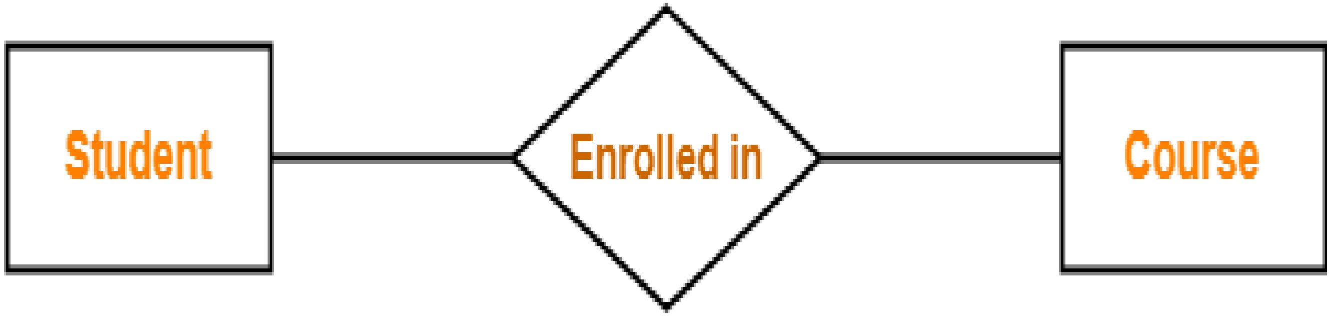
Employee entity

- **Employee no**
- Employee surname
- Employee forename
- Employee DOB
- Employee dept number
- Manager no * (this is the employee no of the employee's manager)



2. Binary Relationship Set-

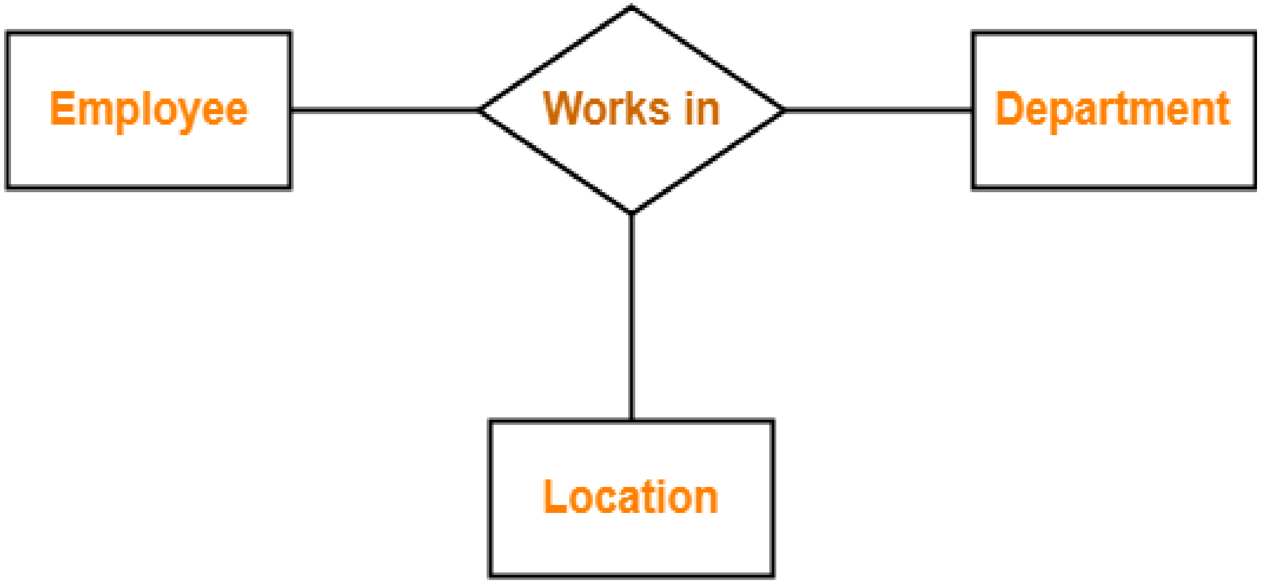
- Binary relationship set is a relationship set where two entity sets participate in a relationship set.



Binary Relationship Set

3. Ternary Relationship Set-

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



Ternary Relationship Set

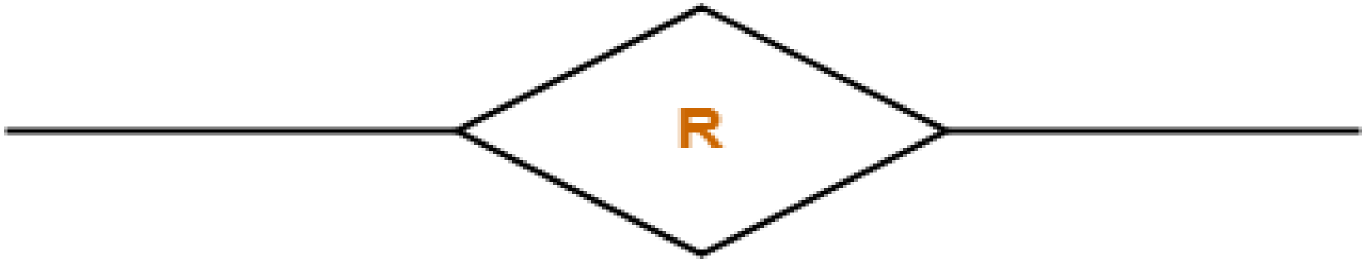
Cardinality in ER Diagram

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

- There are 4 types of cardinality ratios-
  - Many-to-Many cardinality (m:n)
  - Many-to-One cardinality (m:1)
  - One-to-Many cardinality (1:n)
  - One-to-One cardinality (1:1)

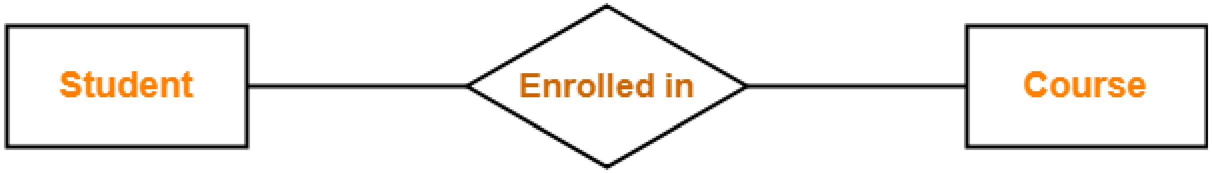
1. Many-to-Many Cardinality

- 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.

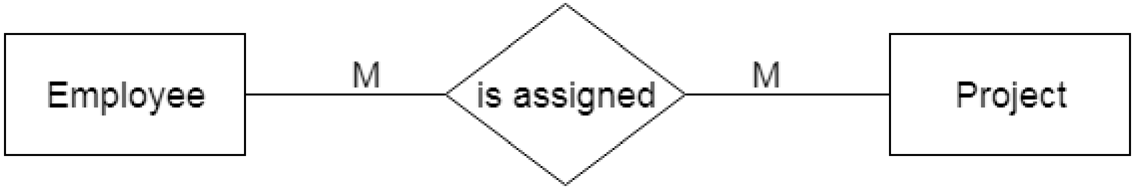


Cardinality Ratio = m : n

- 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.

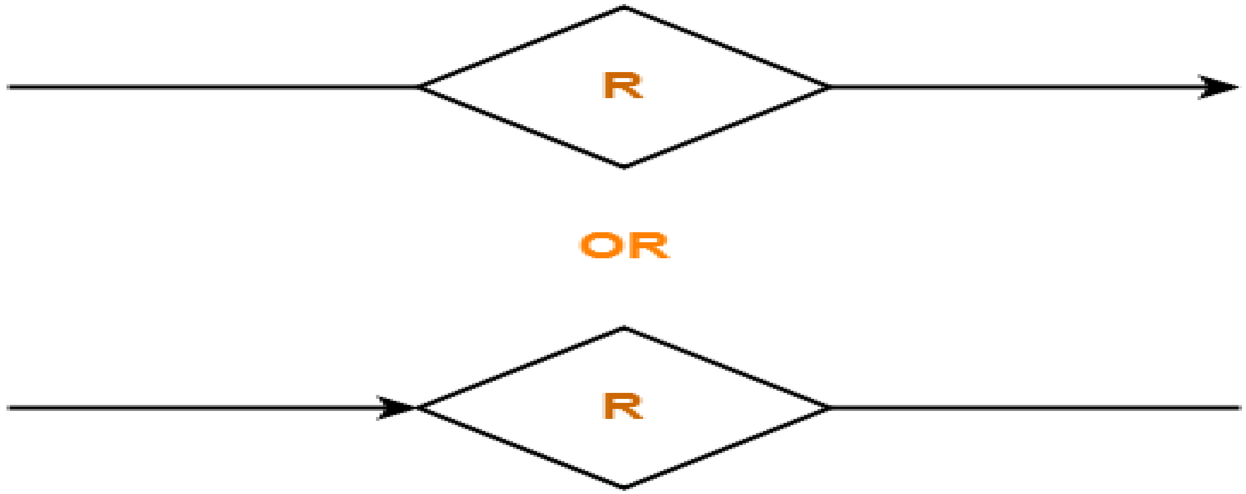


Many to Many Relationship



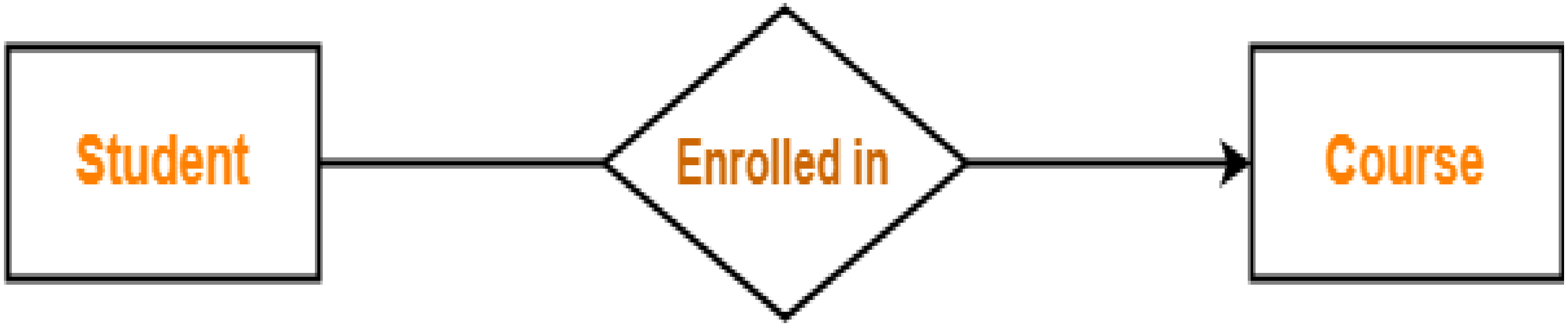
2. Many-to-One Cardinality

- 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.



Cardinality Ratio = m : 1

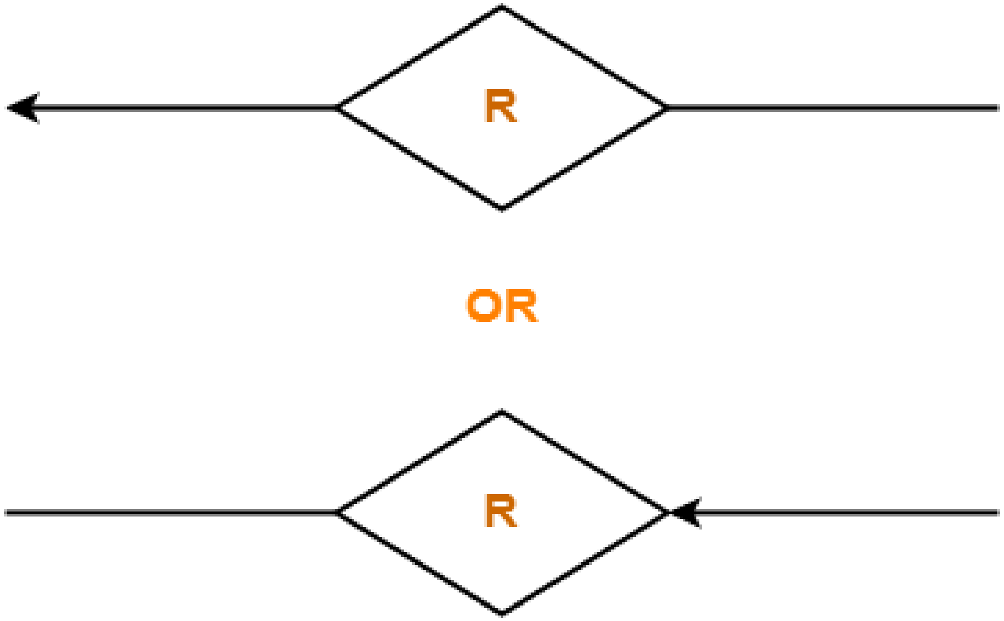
- One student can enroll in at most one course.
- One course can be enrolled by any number (zero or more) of students.



Many to One Relationship

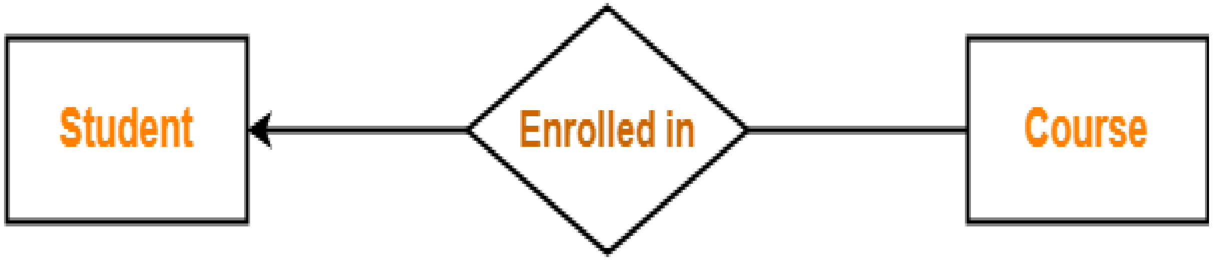
3. One-to-Many Cardinality

- 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.

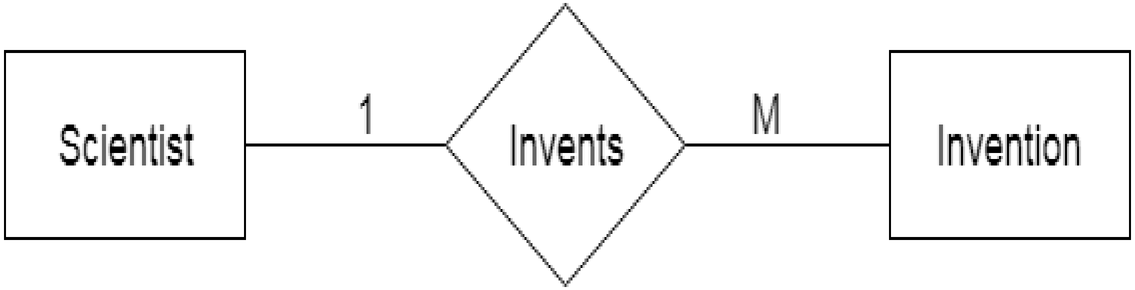


Cardinality Ratio = 1 : n

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



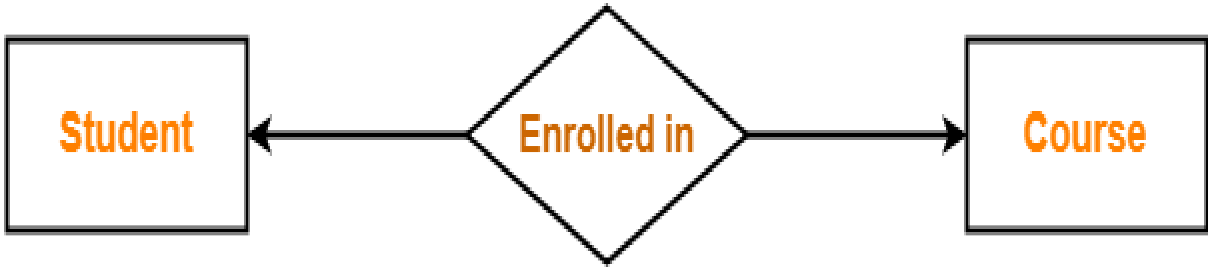
One to Many Relationship



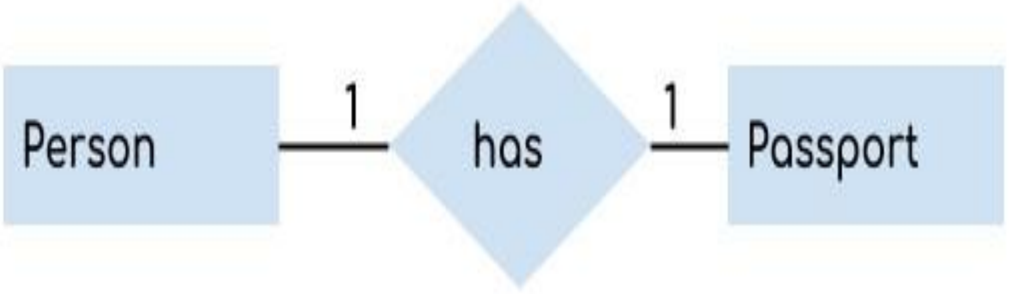
4. One-to-One Cardinality

- 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.

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



One to One Relationship



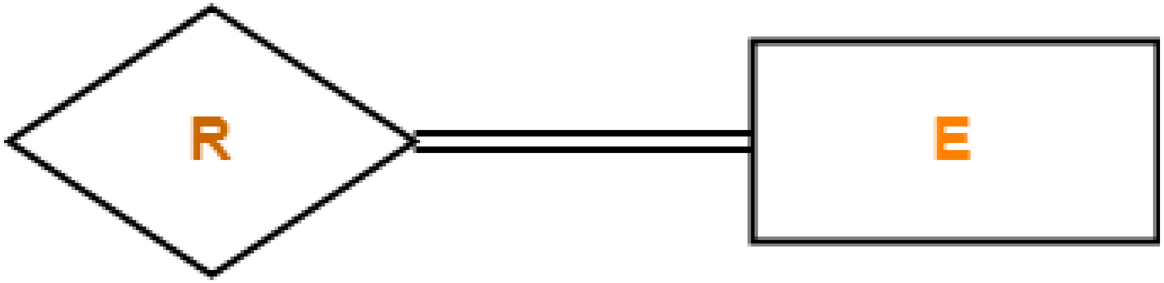
Participation Constraints

- Participation constraints define the least number of relationship instances in which an entity must compulsorily participate.

- There are two types of participation constraints-
  - Total participation
  - Partial participation

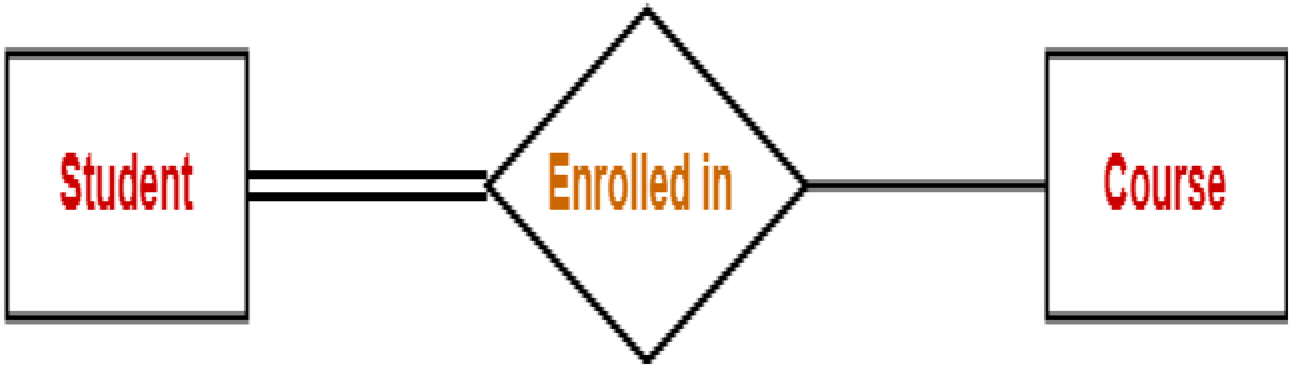
1. Total Participation-

- 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.



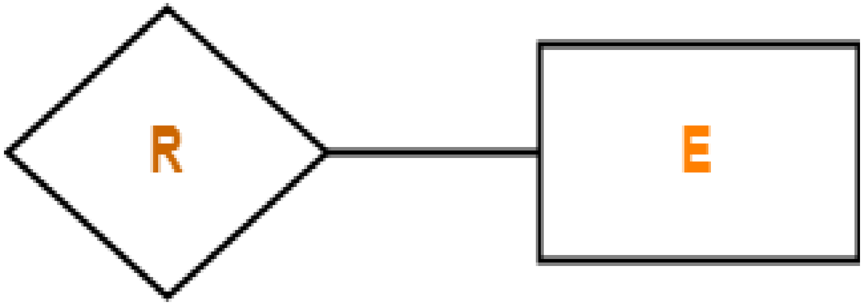
Total Participation

- 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.



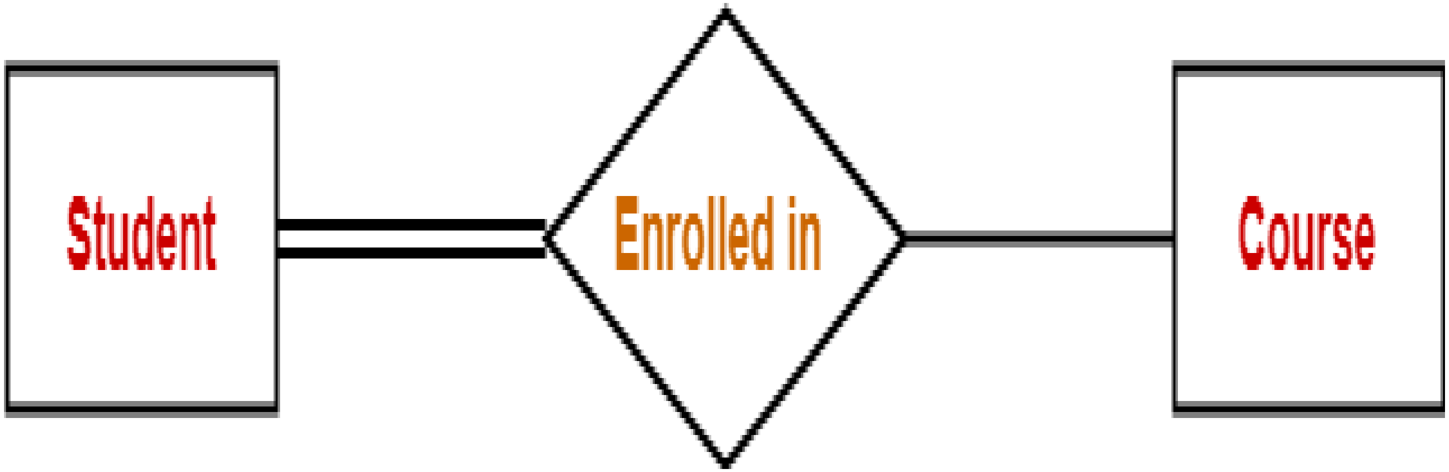
2. Partial Participation

- 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

- 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-

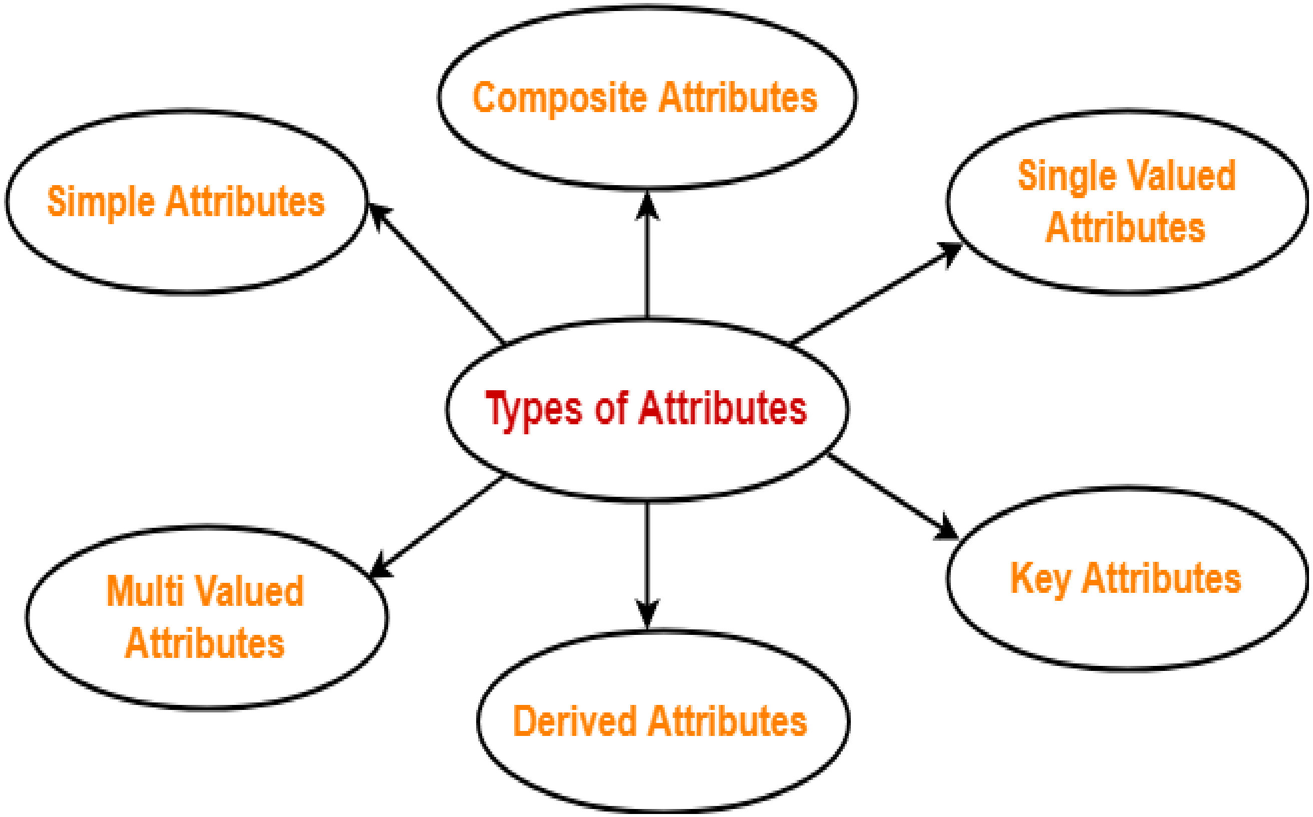
- Minimum cardinality tells whether the participation is partial or total.
  - If minimum cardinality = 0, then it signifies partial participation.
  - If minimum cardinality = 1, then it signifies total participation.
- Maximum cardinality tells the maximum number of entities that participates in a relationship set.

Types of Attributes

- Attributes are the descriptive properties which are owned by each entity of an **Entity Set.**
- There exist a specific domain or set of values for each attribute from where the attribute can take its values.

Types of Attributes

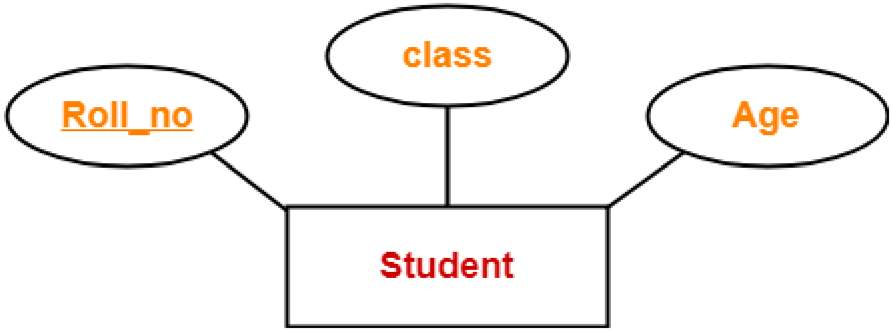
- In ER diagram, attributes associated with an entity set may be of the following types-



1. Simple Attributes-

- Simple attributes are those attributes which can not be divided further.

Example

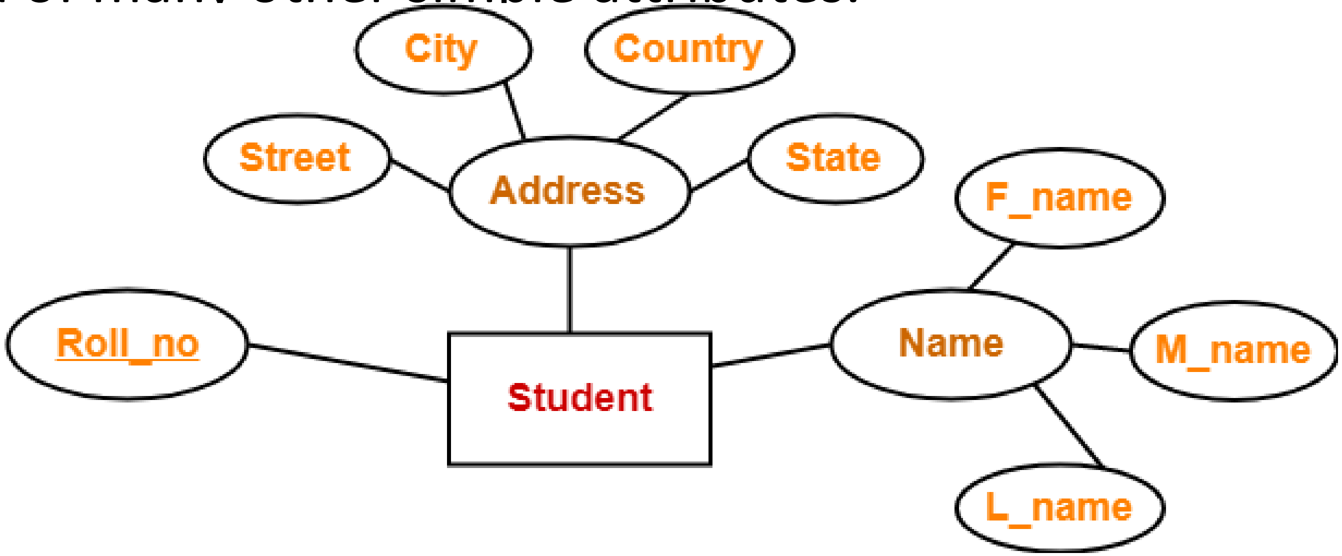


- Here, all the attributes are simple attributes as they can not be divided further.

2. Composite Attributes

- Composite attributes are those attributes which are composed of many other simple attributes.

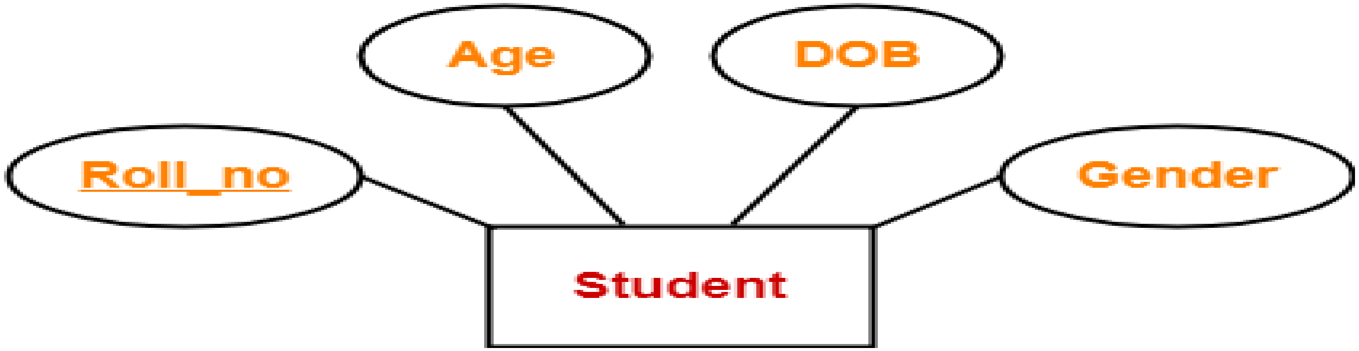
Example-



- Here, the attributes “Name” and “Address” are composite attributes as they are composed of many other simple attributes.

3. Single Valued Attributes

- Single valued attributes are those attributes which can take only one value for a given entity from an entity set.

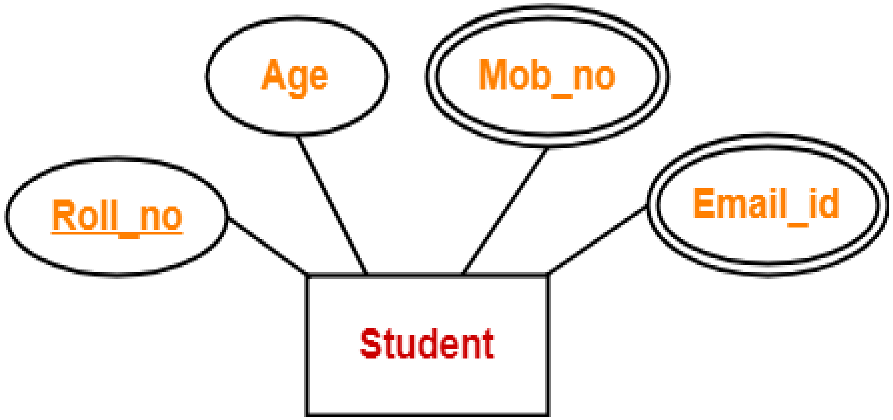


- Here, all the attributes are single valued attributes as they can take only one specific value for each entity.

4. Multi Valued Attributes

- Multi valued attributes are those attributes which can take more than one value for a given entity from an entity set.

<u>Example-</u>

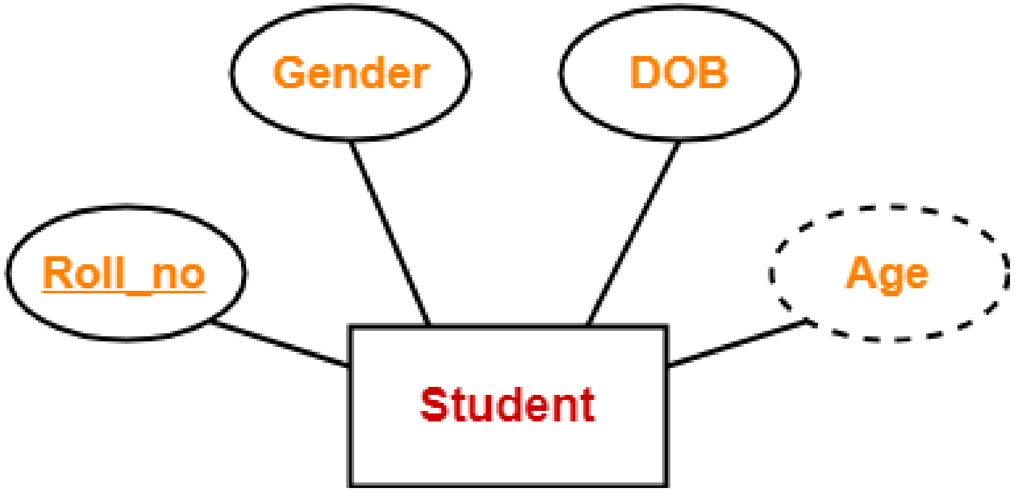


- Here, the attributes “Mob_no” and “Email_id” are multi valued attributes as they can take more than one values for a given entity.

5. Derived Attributes

- Derived attributes are those attributes which can be derived from other attribute(s).

Example-

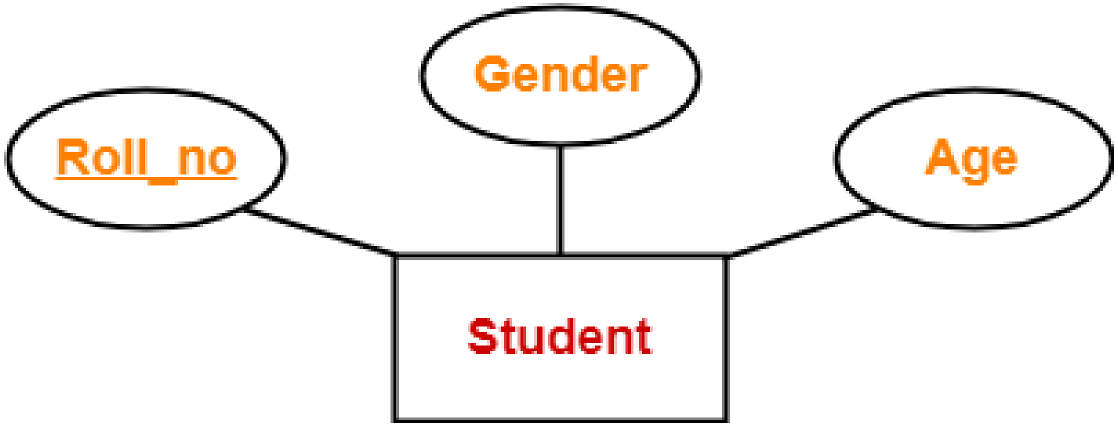


- Here, the attribute “Age” is a derived attribute as it can be derived from the attribute “DOB”.

6. Key Attributes-

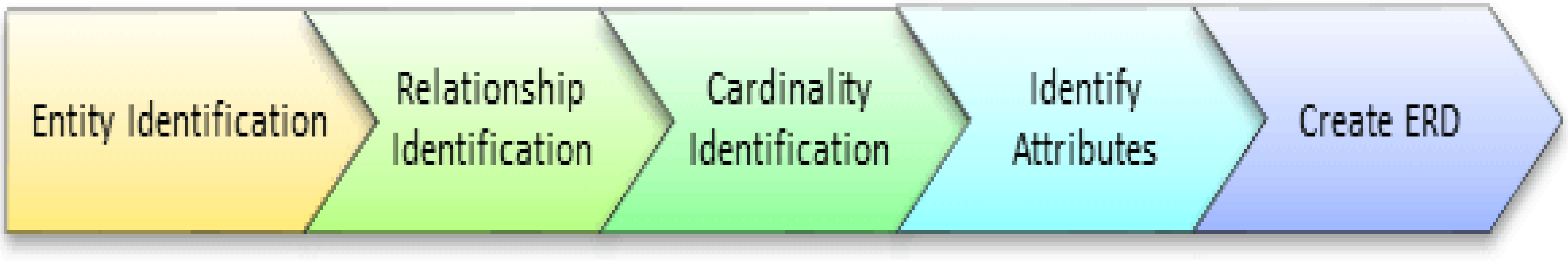
- Key attributes are those attributes which can identify an entity uniquely in an entity set.

Example-



- Here, the attribute “Roll_no” is a key attribute as it can identify any student uniquely.

Steps to Create an ERD



Entity Identification

Relationship Identification

Cardinality Identification

Identify Attributes

Create ERD

Steps to Create an ERD

- In a university, a Student enrolls in Courses.
- A student must be assigned to at least one or more Courses.
- Each course is taught by a single Professor.
- To maintain instruction quality, a Professor can deliver only one course

Steps to Create an ERD

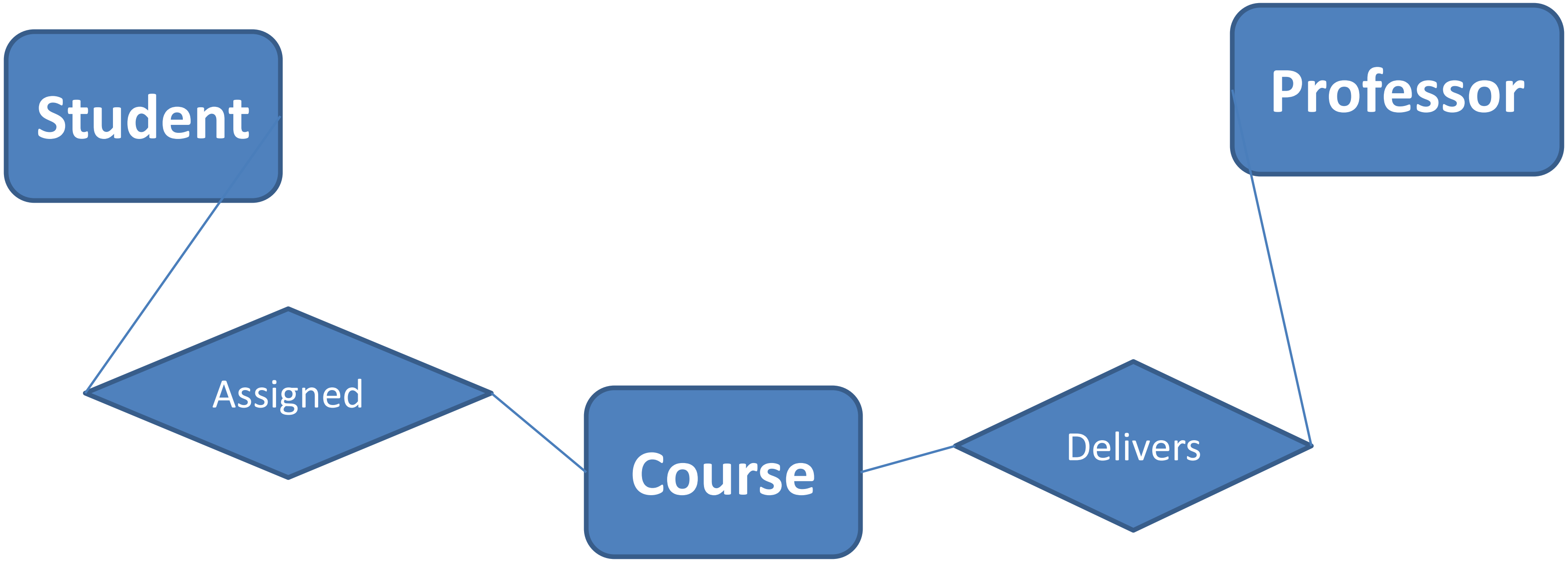
Step 1) Entity Identification

- We have three entities
  - Student
  - Course
  - Professor

Steps to Create an ERD

Step 2) Relationship Identification

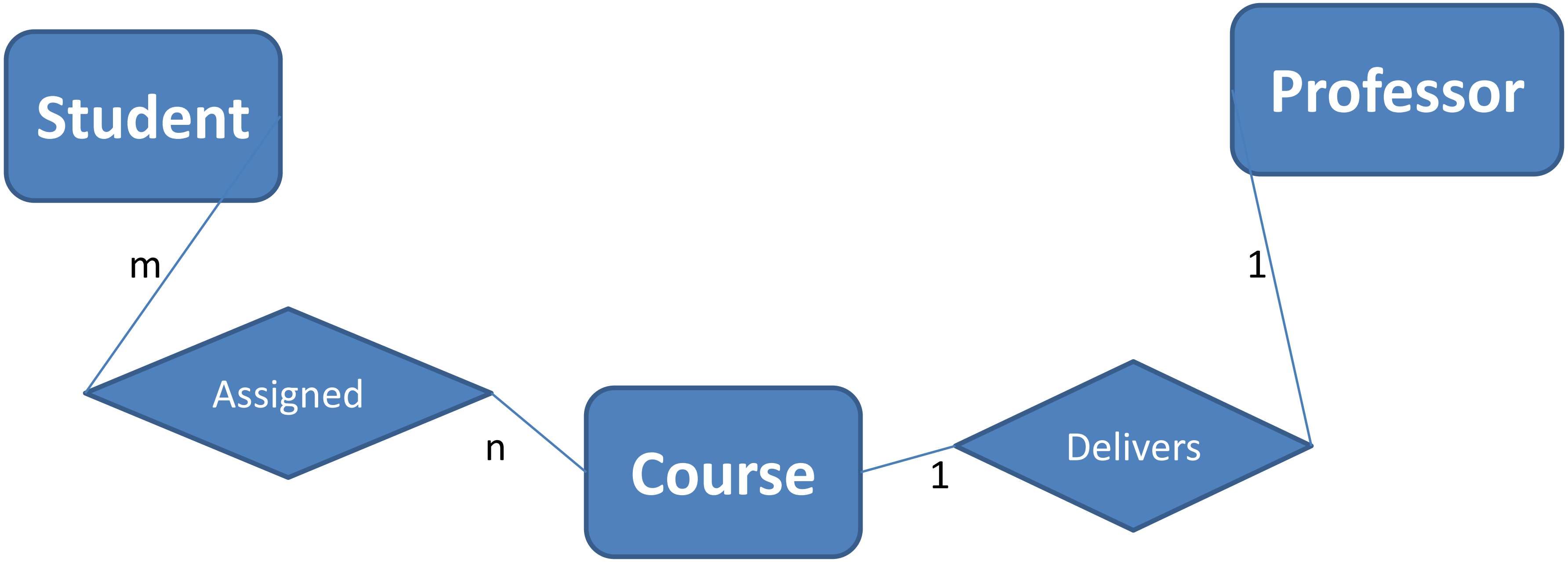
- We have the following two relationships
  - The student is **assigned** a course
  - Professor **delivers** a course



Steps to Create an ERD

Step 3) Cardinality Identification

- For them problem statement we know that,
  - A student can be assigned **multiple** courses
  - A Professor can deliver only **one** course



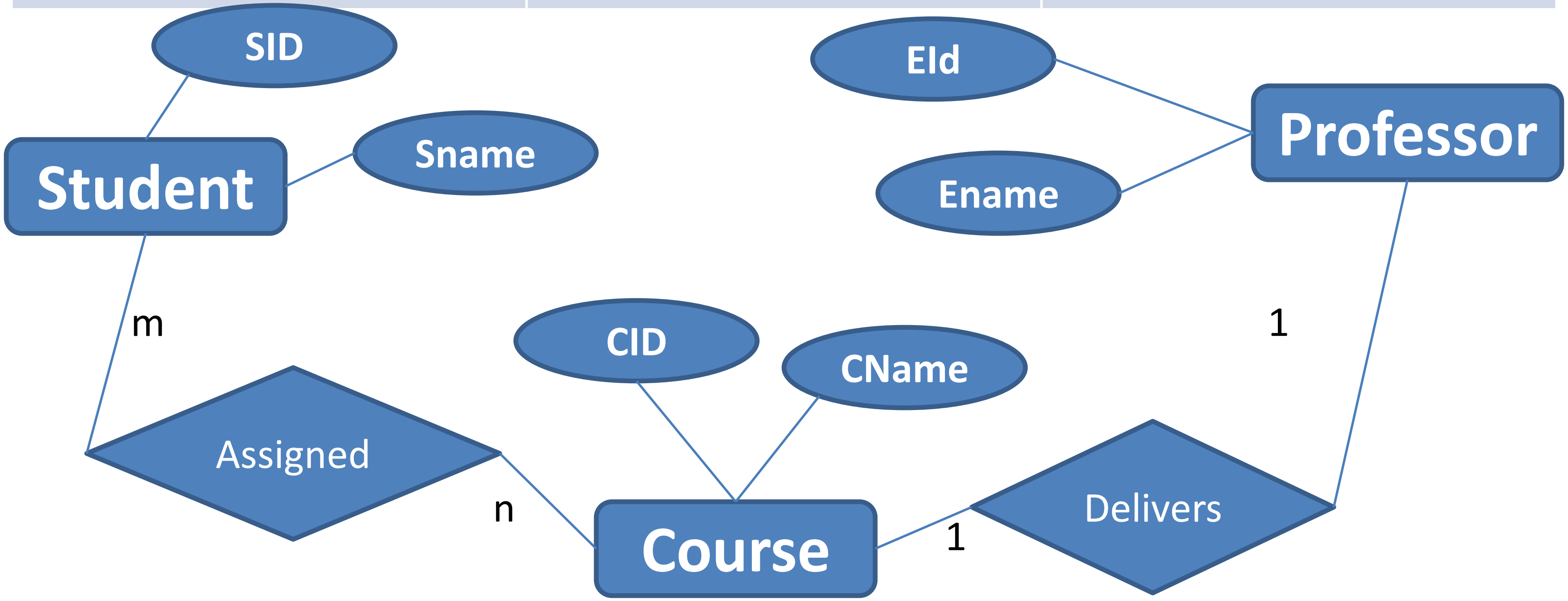
Steps to Create an ERD

Step 4) Identify Attributes

- You need to study the files, forms, reports, data currently maintained by the organization to identify attributes.
- You can also conduct interviews with various stakeholders to identify entities.
- Initially, it's important to identify the attributes without mapping them to a particular entity.
- Once, you have a list of Attributes, you need to map them to the identified entities.
- Once the mapping is done, identify the primary Keys.
- If a unique key is not readily available, create one.

Steps to Create an ERD

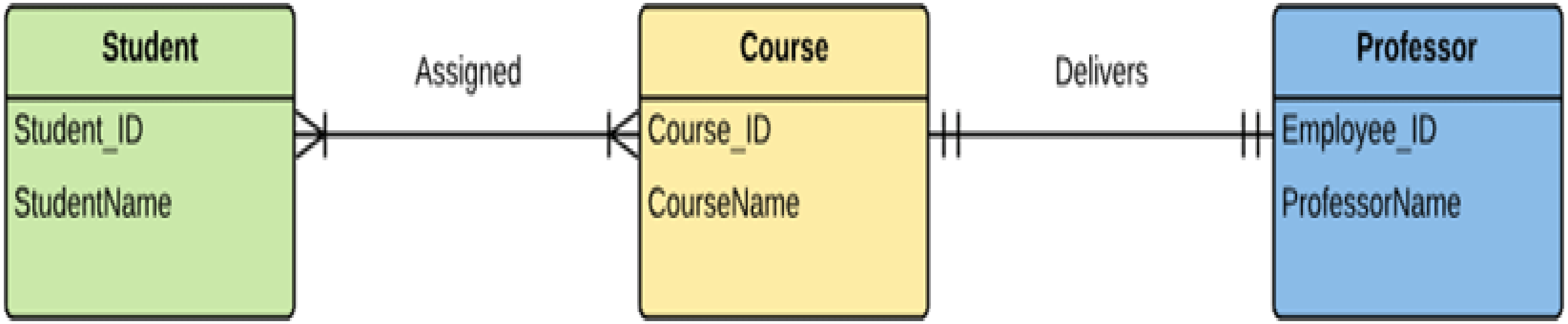
Entity	Primary Key	Attribute
Student	Student_ID	StudentName
Professor	Employee_ID	ProfessorName
Course	Course_ID	CourseName



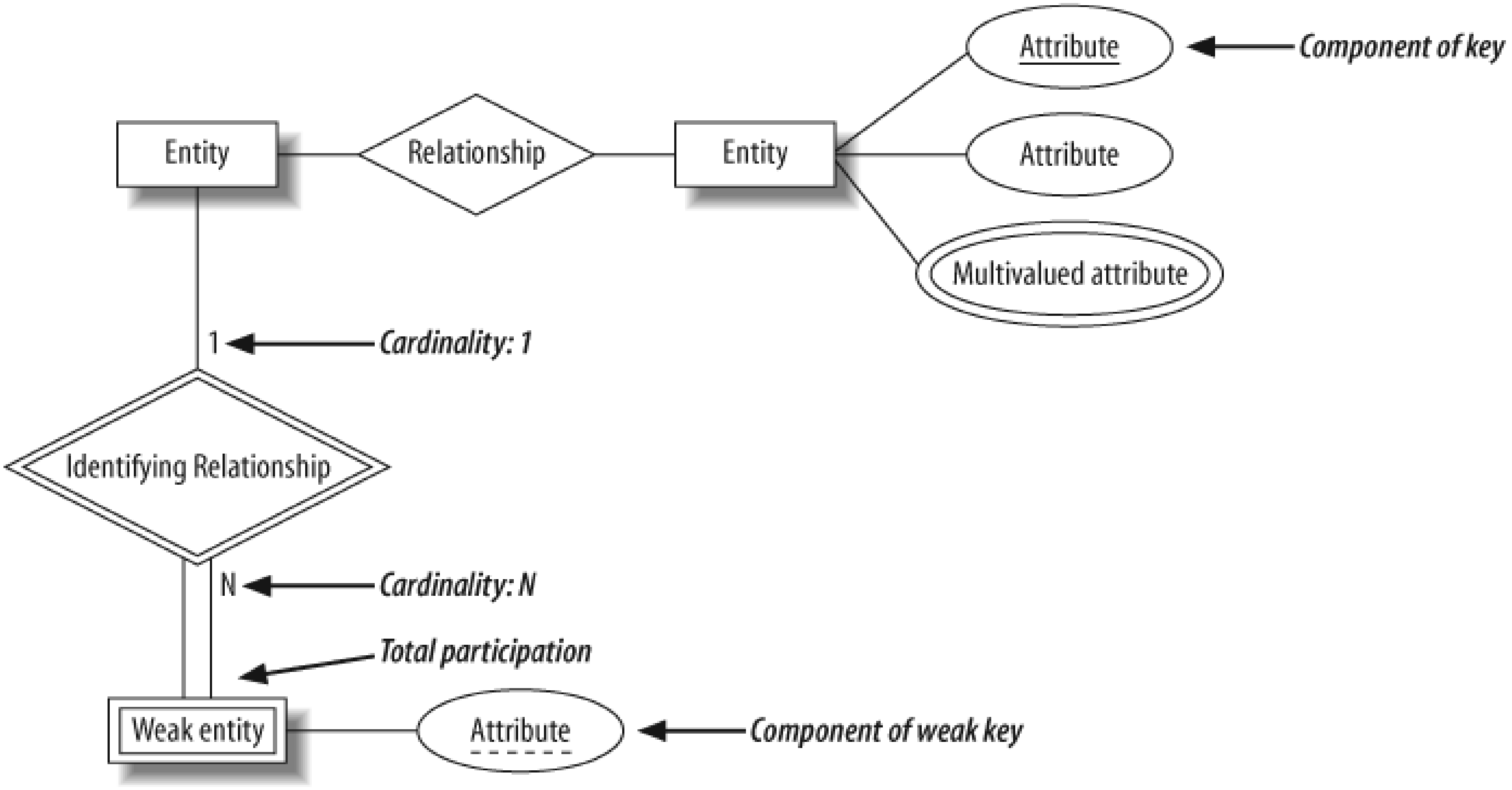
Steps to Create an ERD

Step 5) Create the ERD

- A more modern representation of ERD Diagram



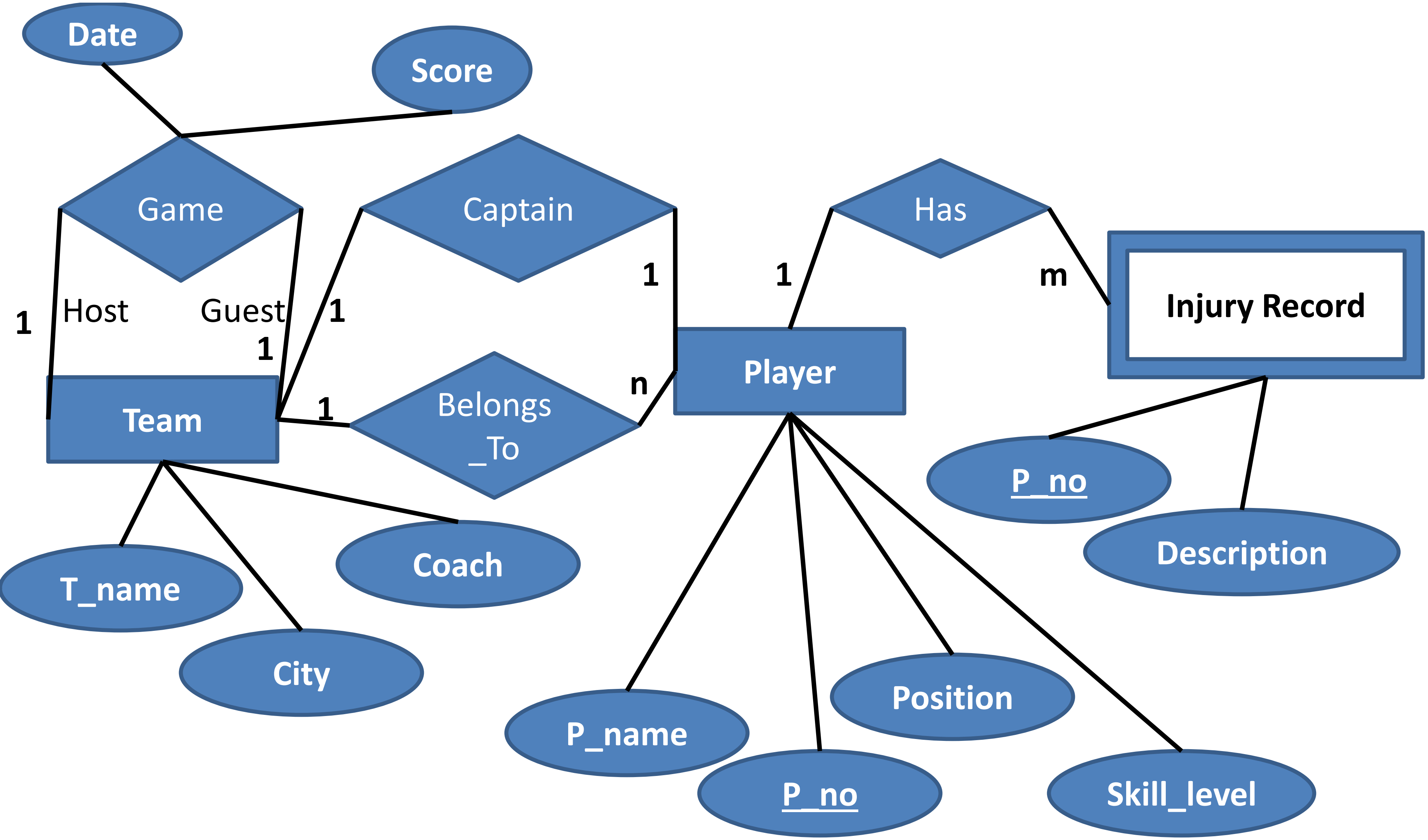
Steps to Create an ERD



ER diagram example

- Suppose you are given the following requirements for a simple database for the National
- Hockey League (NHL):
  - the NHL has many teams,
  - each team has a name, a city, a coach, a captain, and a set of players,
  - each player belongs to only one team,
  - each player has a name, a position (such as left wing or goalie), a skill level, and a set of injury records,
  - a team captain is also a player,
  - a game is played between two teams (referred to as host_team and guest_team) and has a date (such as May 11th, 1999) and a score (such as 4 to 2).

- Entities:
  - Team(t_name, city, coach)
  - Player(p_name, position, skill_level)
  - Injury record (**Weak entity, depend on player**)
- Relationships:
  - Each team has **a captain** which is also a player
  - Each team **has** many player
  - A **game** is played between two teams(host and guest), and has date and score(attributes)
  - A player has injury record


Date
Score
Game
Captain
Has
1
Host
Guest
1
1
1
1
m
Injury Record
Player
n
Team
1
Belongs
_To
P_no
Description
T_name
Coach
City
Position
P_name
P_no
Skill_level

Example 2

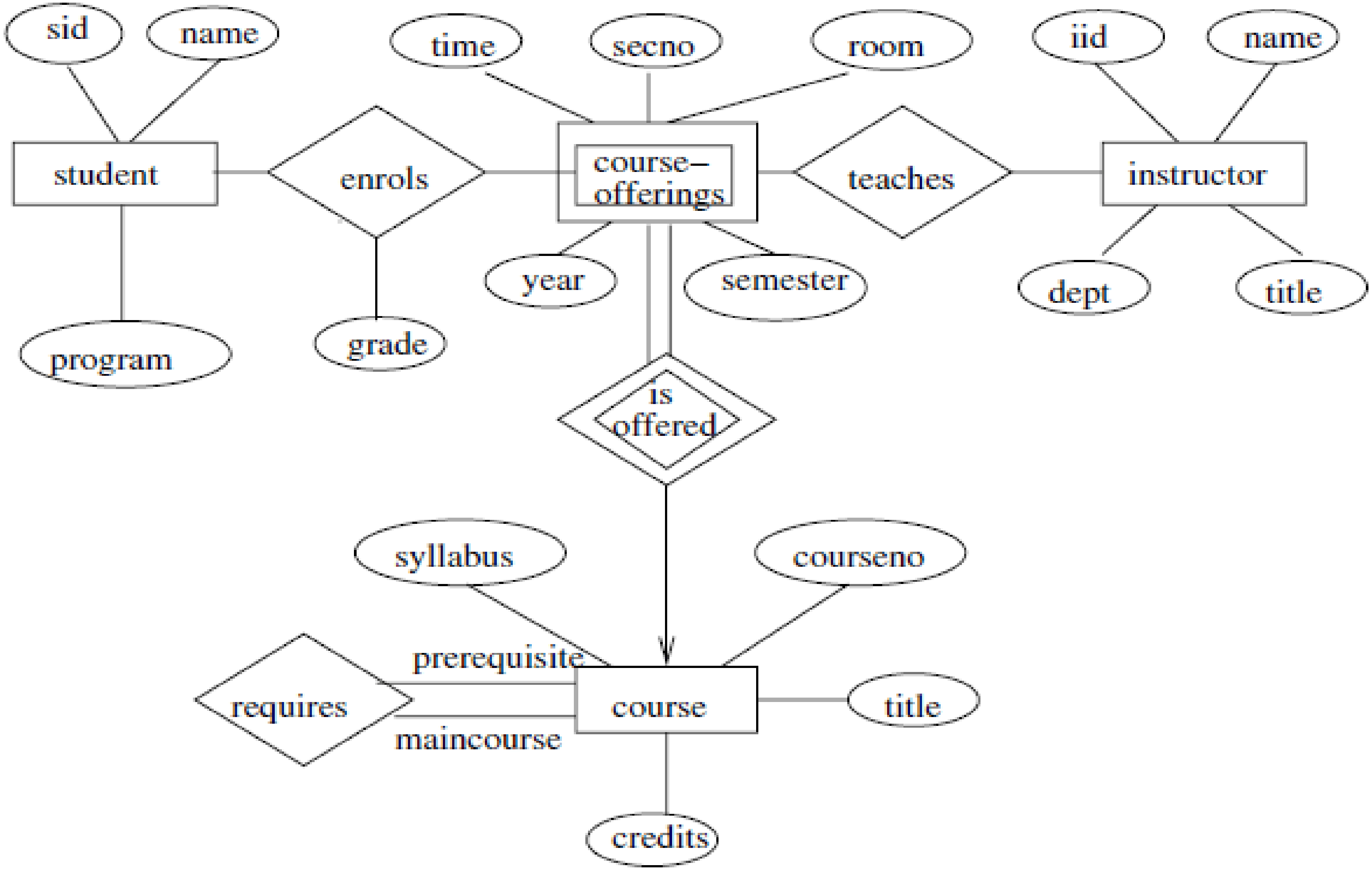
- A university registrar’s office maintains data about the following entities:
  - courses, including number, title, credits, syllabus, and prerequisites;
  - course offerings, including course number, year, semester, section number, instructor(s), timings, and classroom;
  - students, including student-id, name, and program;
  - instructors, including identification number, name, department, and title.
- Further, the enrollment of students in courses and grades awarded to students in each course they are enrolled for must be appropriately modeled.
- Construct an E-R diagram for the registrar’s office.
- Document all assumptions that you make about the mapping constraints.

Entities

- Student(sid, name, program)
- Course(C_number, title, credits, syllabus)
- course offerings(c_number, year, semester, section_number, timings, and classroom)
- Instructor(iid, name, department, title)

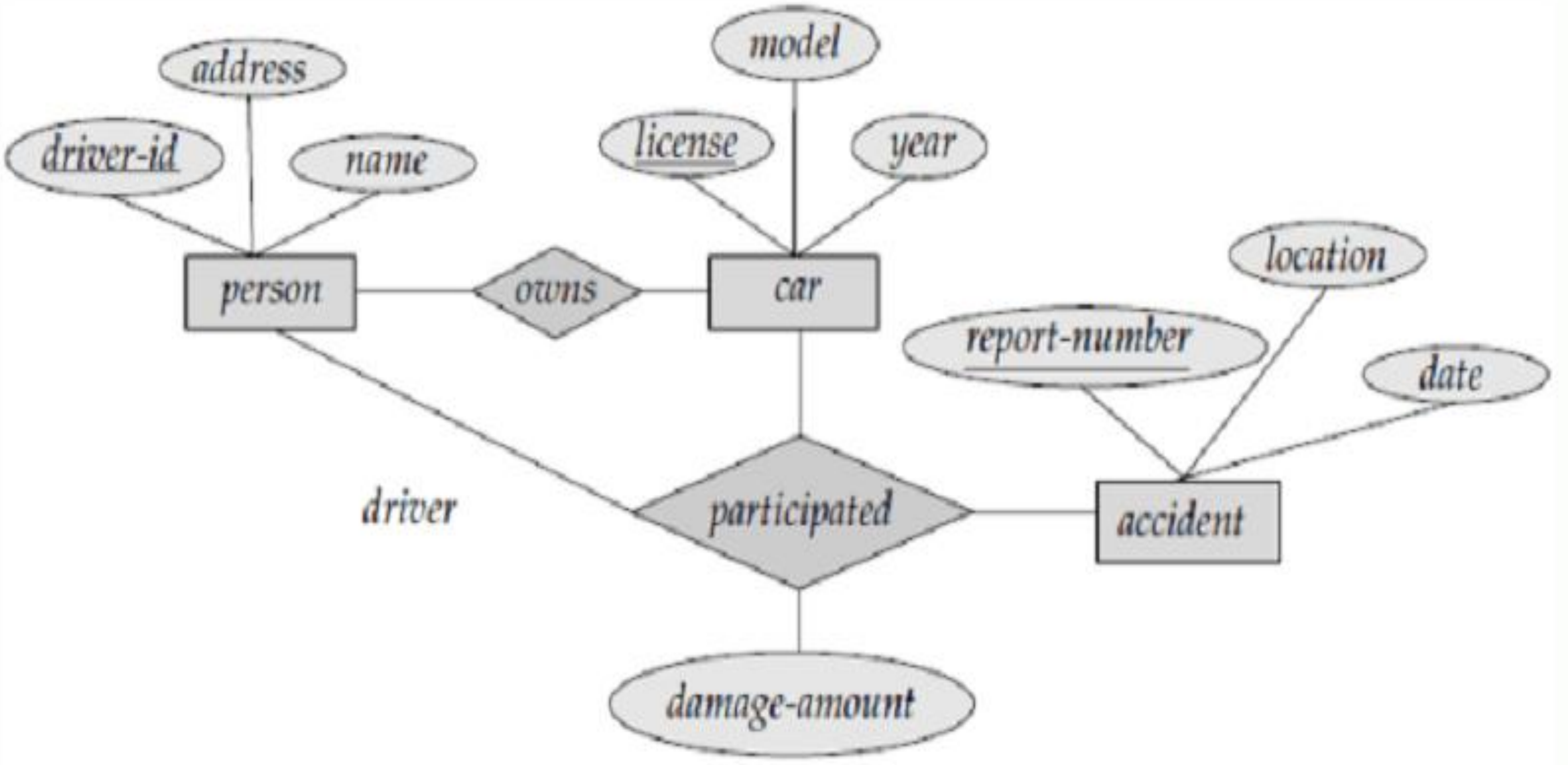
Relationships

- Students enrolls in course offerings, then grade is allotted.
- Instructor teaches course offerings.
- A course is offered Course offerings
- A main course required A prerequisite course.


sid
name
time
secno
room
iid
name
student
enrols
course–
offerings
teaches
instructor
year
semester
dept
title
program
grade
is
offered
syllabus
courseno
prerequisite
requires
course
title
maincourse
credits

Example 3

- Construct an E-R diagram for a car-insurance company whose customers own one or more cars each.
- Each car has associated with it zero to any number of recorded accidents.

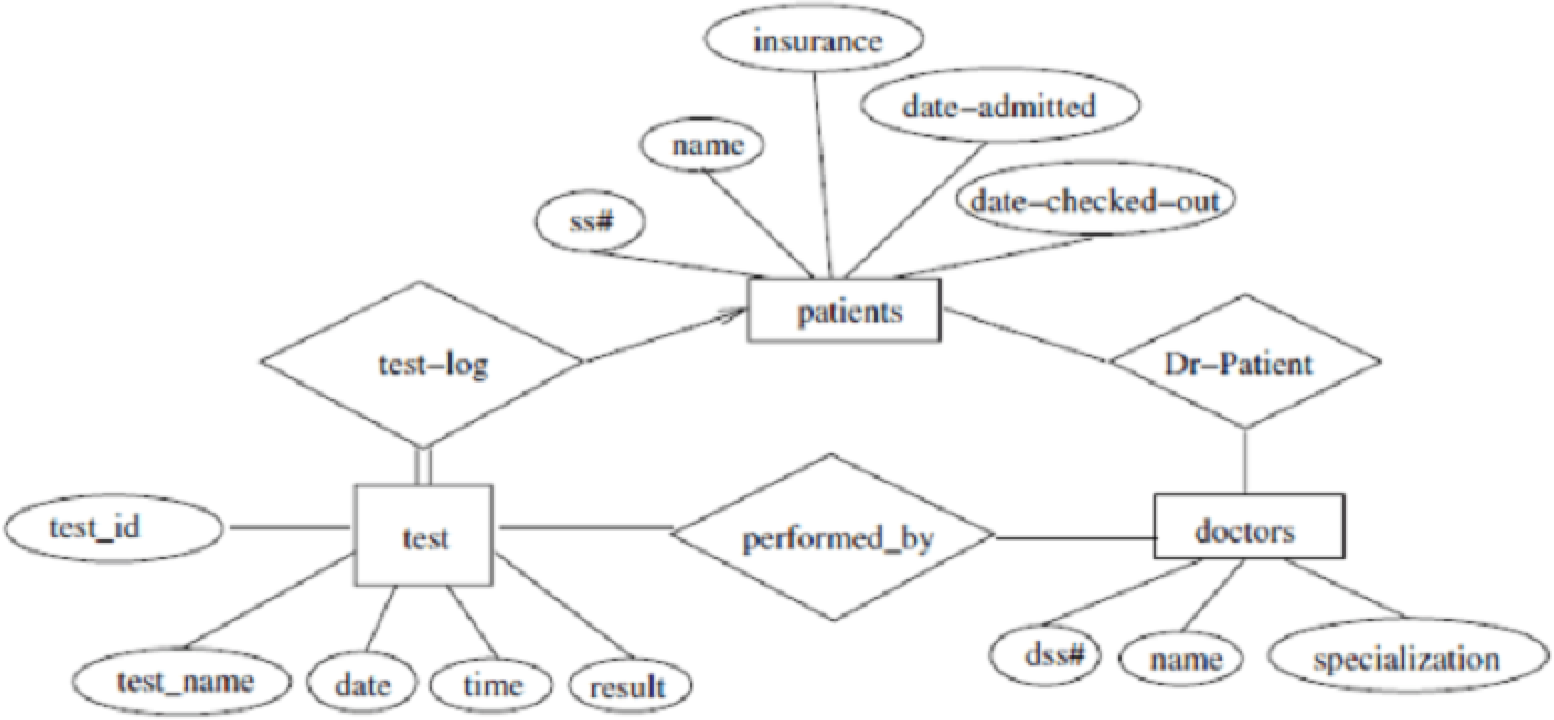


E-R diagram for a Car-insurance company.

- Construct appropriate tables for the above ER Diagram ?
- Car insurance tables:
  - person (driver-id, name, address)
  - car (license, year,model)
  - accident (report-number, date, location)
  - participated(driver-id, license, report-number, damage-amount)

Example 4

- Construct an E-R diagram for a hospital with a set of patients and a set of medical doctors.
- Associate with each patient a log of the various tests and examinations conducted.



E-R diagram for a hospital.

- Construct appropriate tables for the above ER Diagram :
  - Patient(SS#, name, insurance)
  - Physician (name, specialization)
  - Test-log(SS#, test-name, date, time)
  - Doctor-patient (physician-name, SS#)
  - Patient-history(SS#, test-name, date)

Example 5

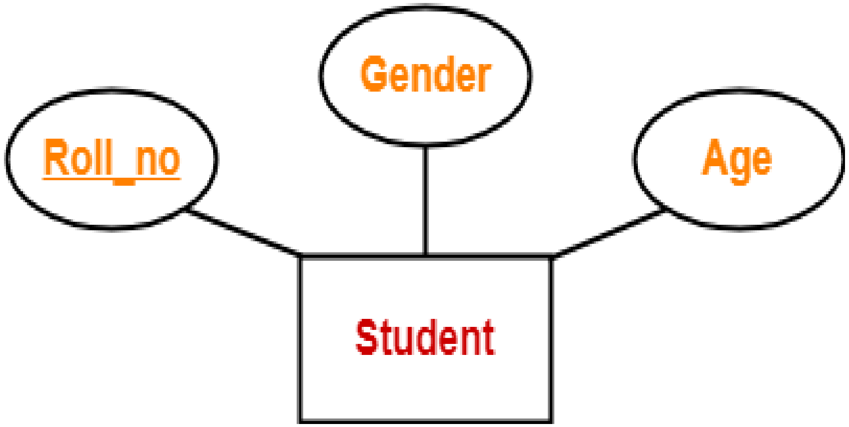
- Draw the E-R diagram which models an online bookstore.

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.

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.

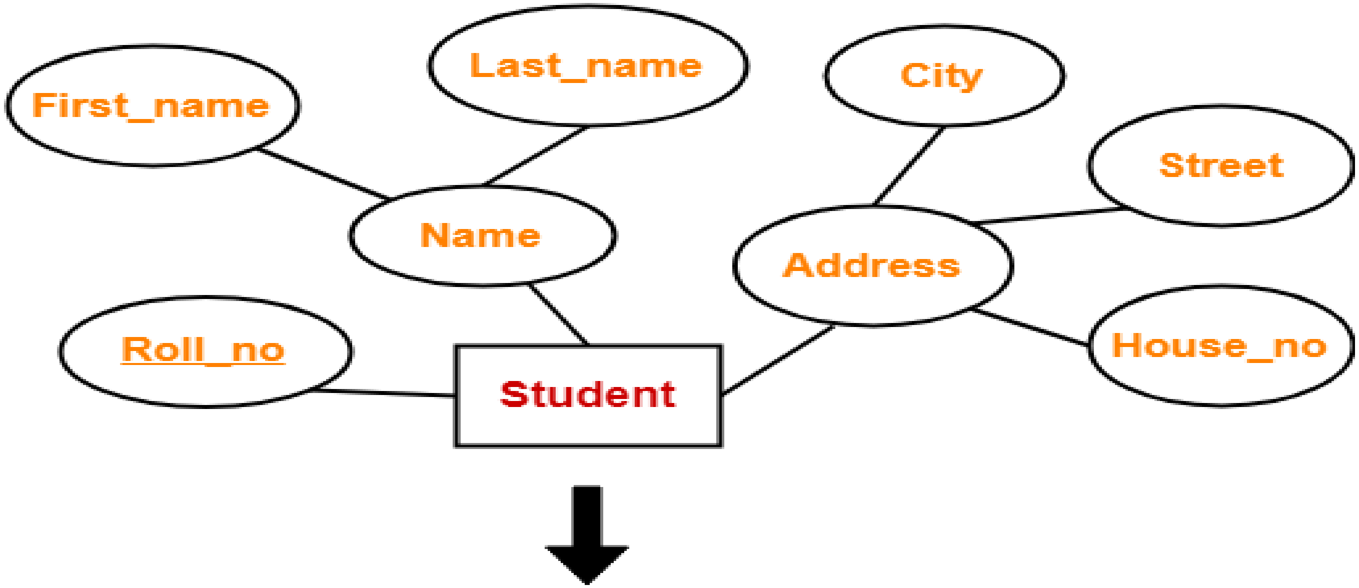


Schema : Student (Roll_no , Gender , Age **)**

Roll_no	Gender	Age

Rule-02: For Strong Entity Set With Composite Attributes

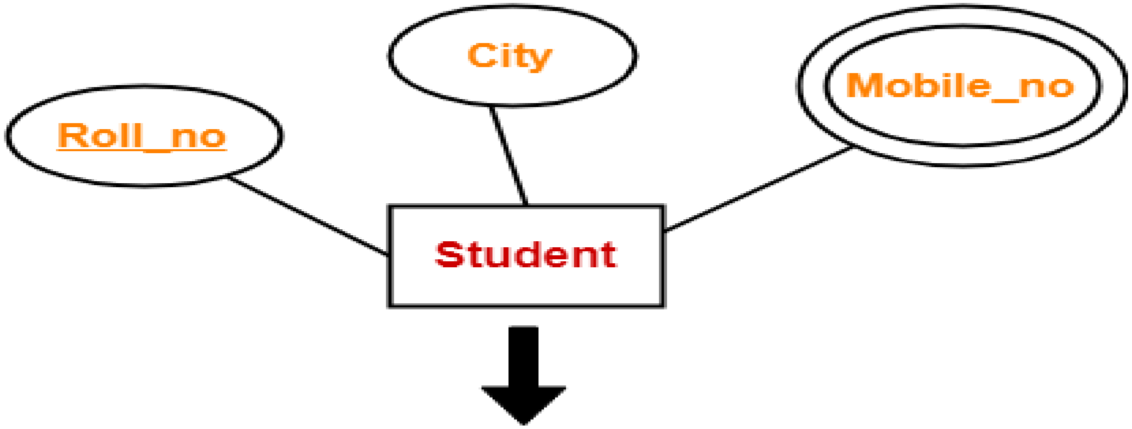
- A strong entity set with any number of composite attributes will require **only one table** in relational model.
- While conversion, simple attributes of the composite attributes are taken into account and not the composite attribute itself.



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.

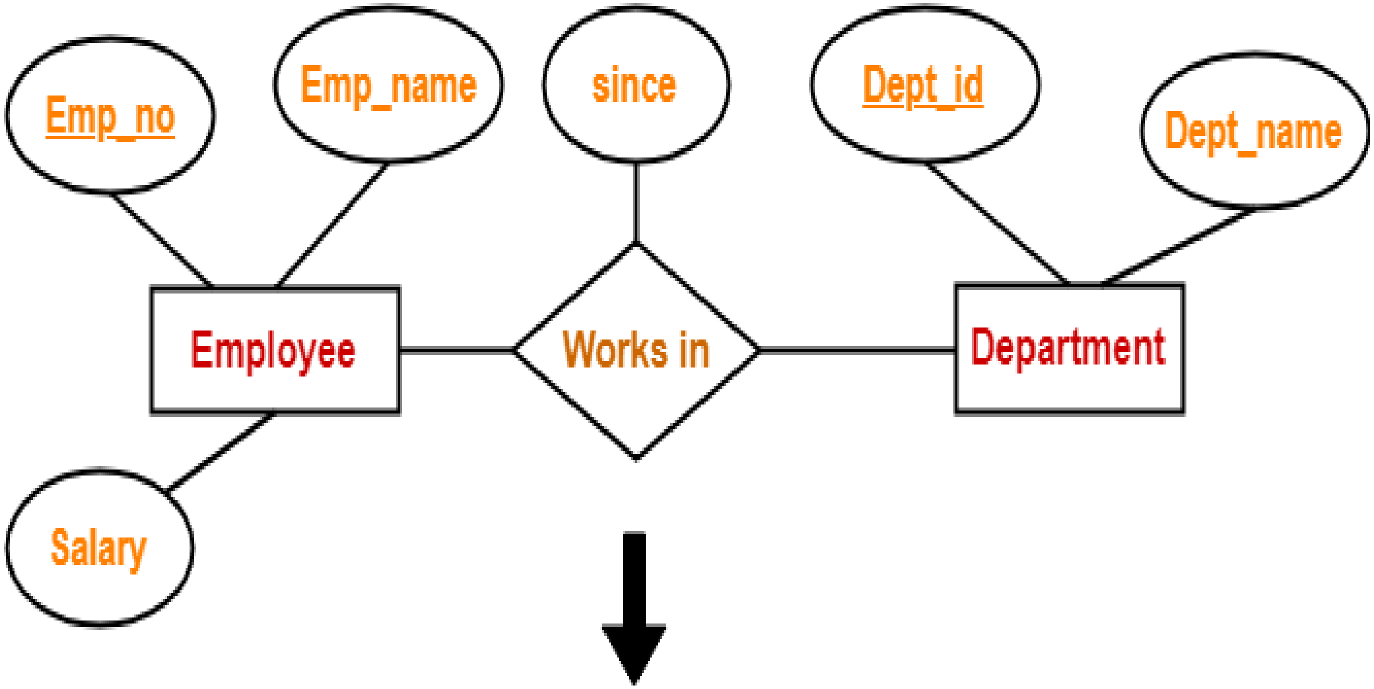


Roll_no	City

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.



<u>Emp_no</u>	<u>Dept_id</u>	Since

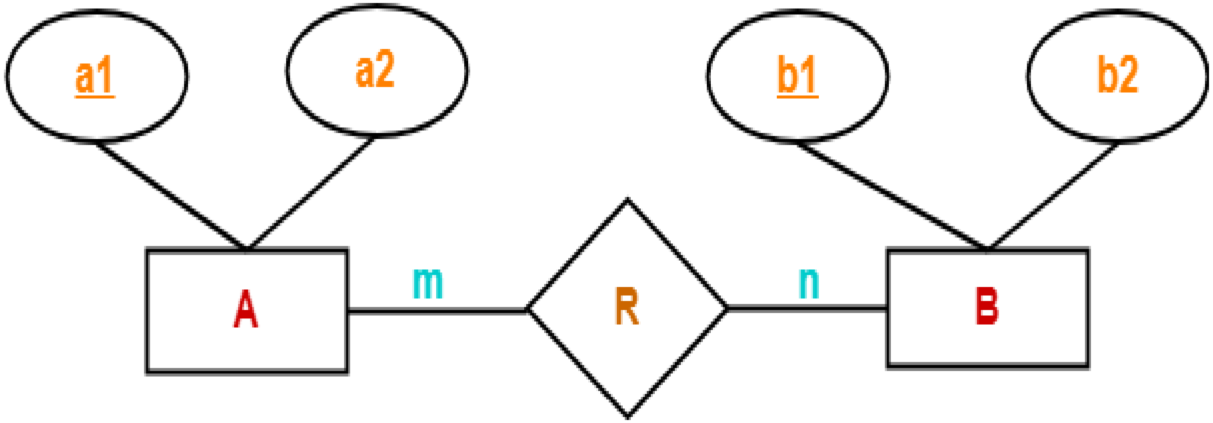
Schema : Works in (<u>Emp_no</u> , <u>Dept_id</u> , since)

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

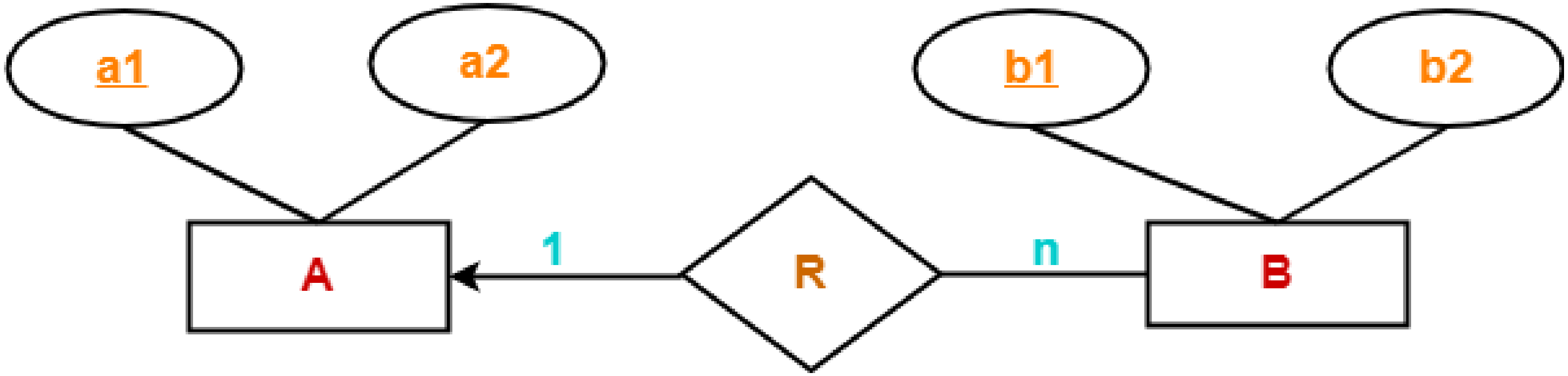
Case-01: For Binary Relationship With Cardinality Ratio m:n



Here, three tables will be required-

- A (a1 , a2)
- R (a1 , b1)
- B (b1 , b2)

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

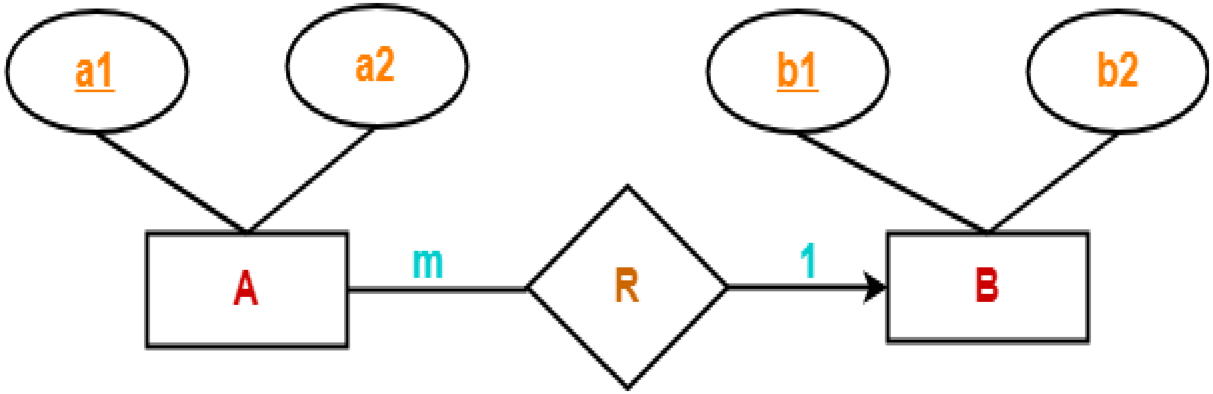


Here, two tables will be required-

- A (<u>a1</u> , a2)
- BR (a1 , <u>b1</u> , b2)

<u>**NOTE-**</u> 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



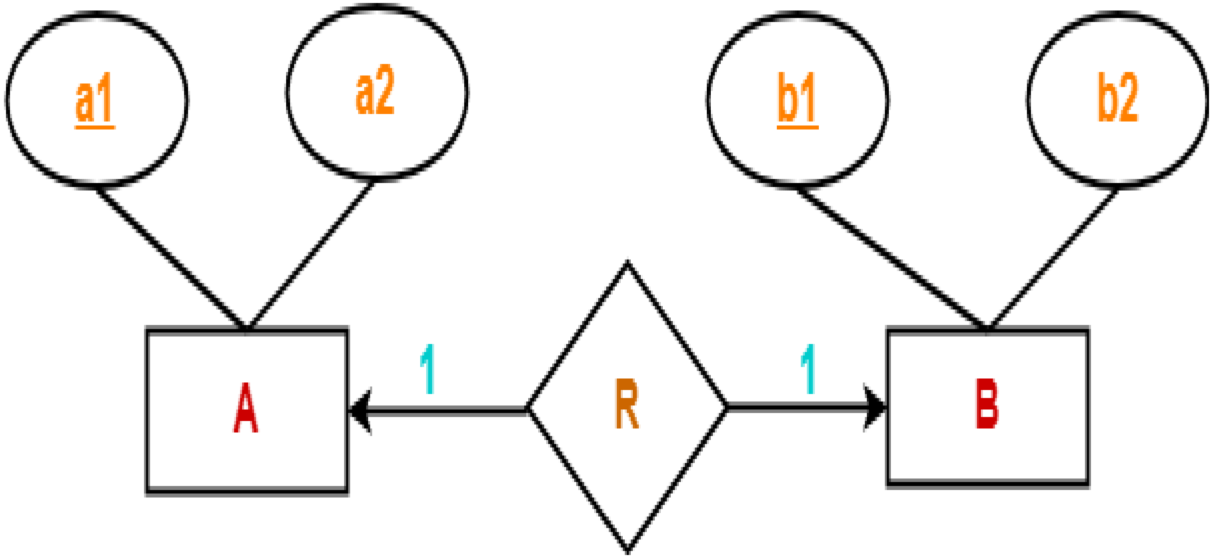
Here, two tables will be required-

- AR (a1 , a2 , b1)
- 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:

AR (a1 , a2 , b1)
B (b1 , b2)

Way-02:

A (a1 , a2)
BR (a1 , b1 , b2)

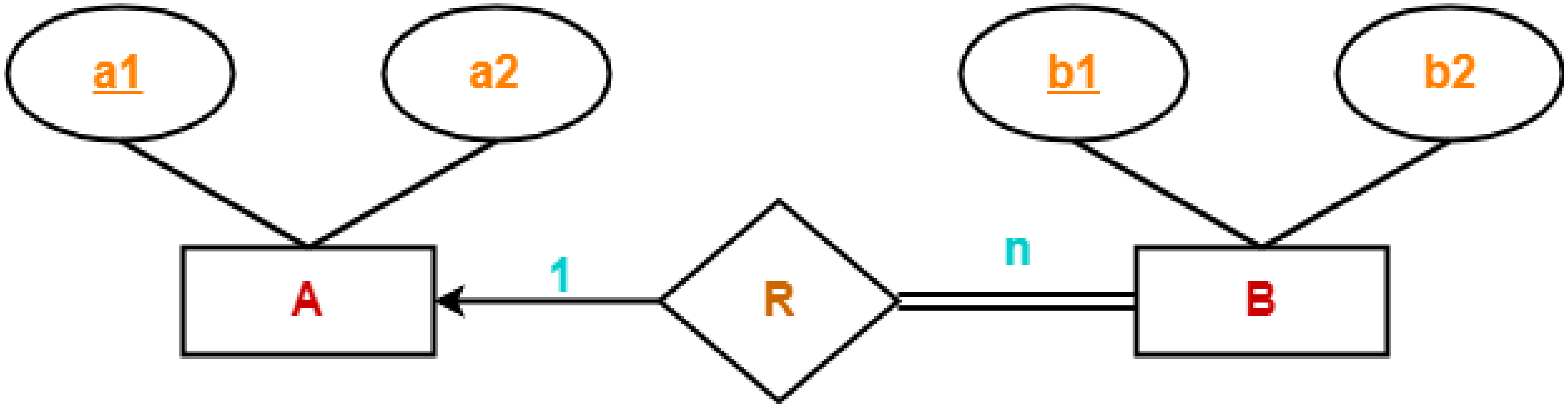
Thumb Rules to Remember

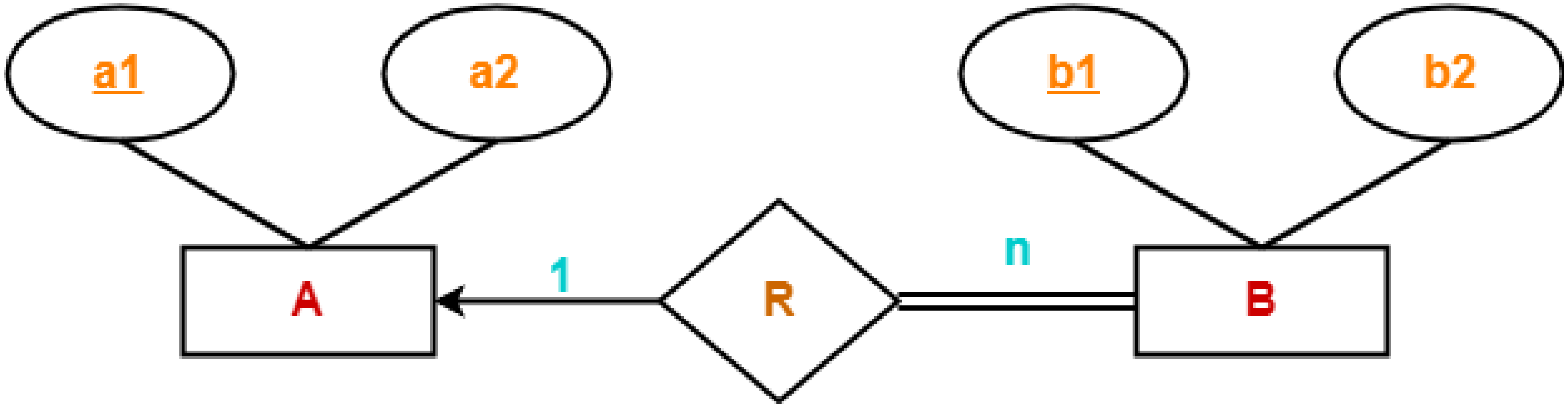
- While determining the minimum number of tables required for binary relationships with given cardinality ratios, following thumb rules must be kept in mind-
  - For binary relationship with cardinality ration m : n , separate and individual tables will be drawn for each entity set and relationship.
  - 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-

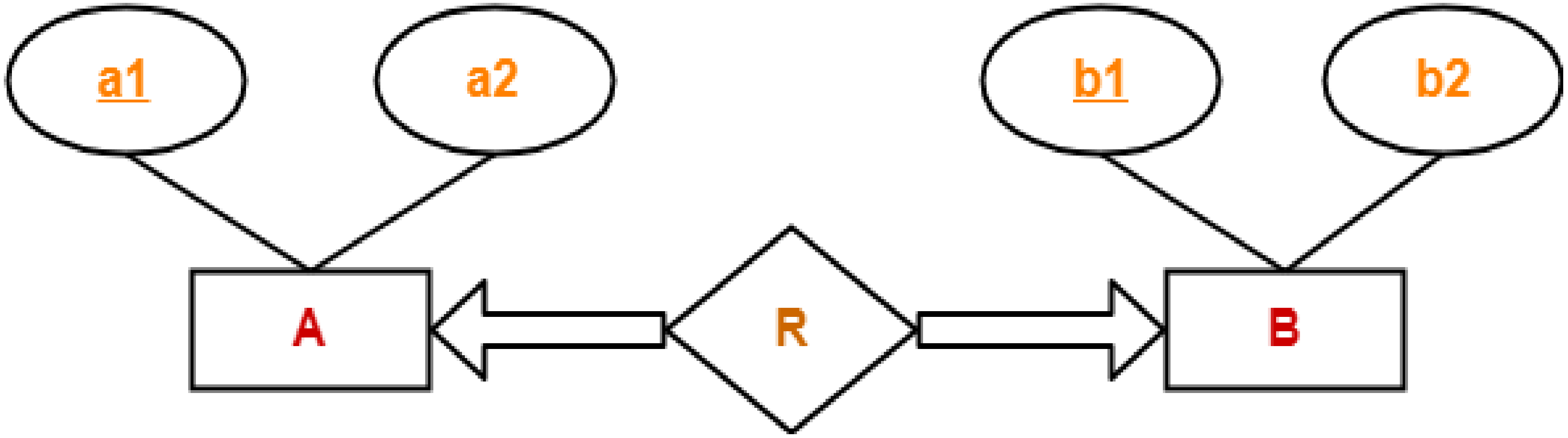
  A (a1 , a2)

  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 (<u>a1</u> , a2 , <u>b1</u> , b2)

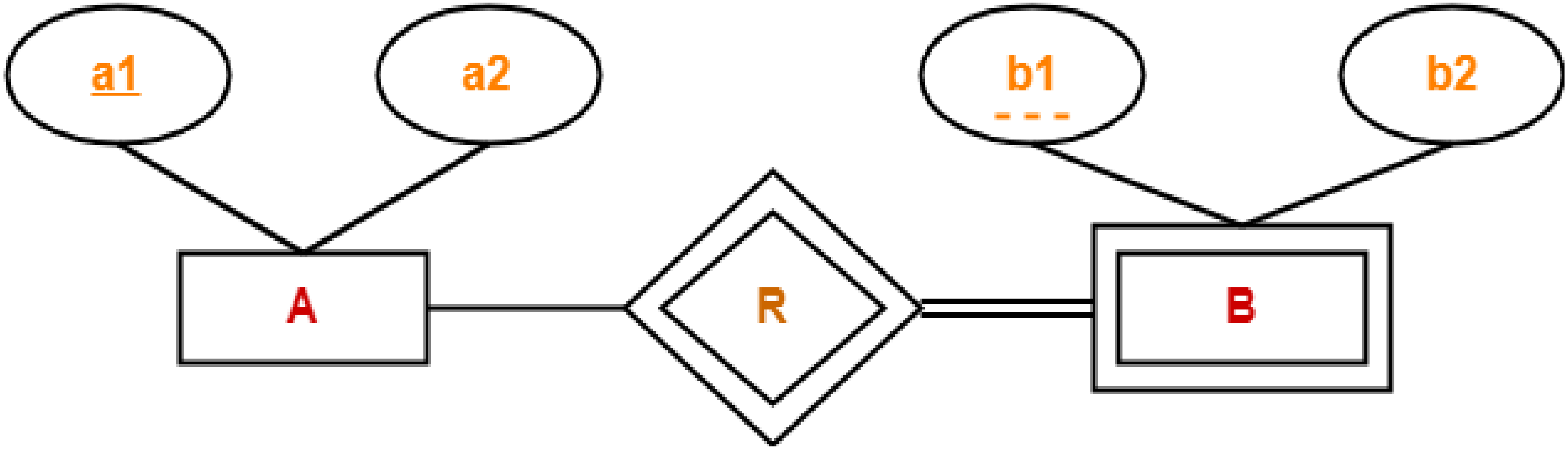


Rule-07: For Binary Relationship With Weak Entity Set

- Weak entity set always appears in association with identifying relationship with total participation constraint.
- Here, two tables will be required-

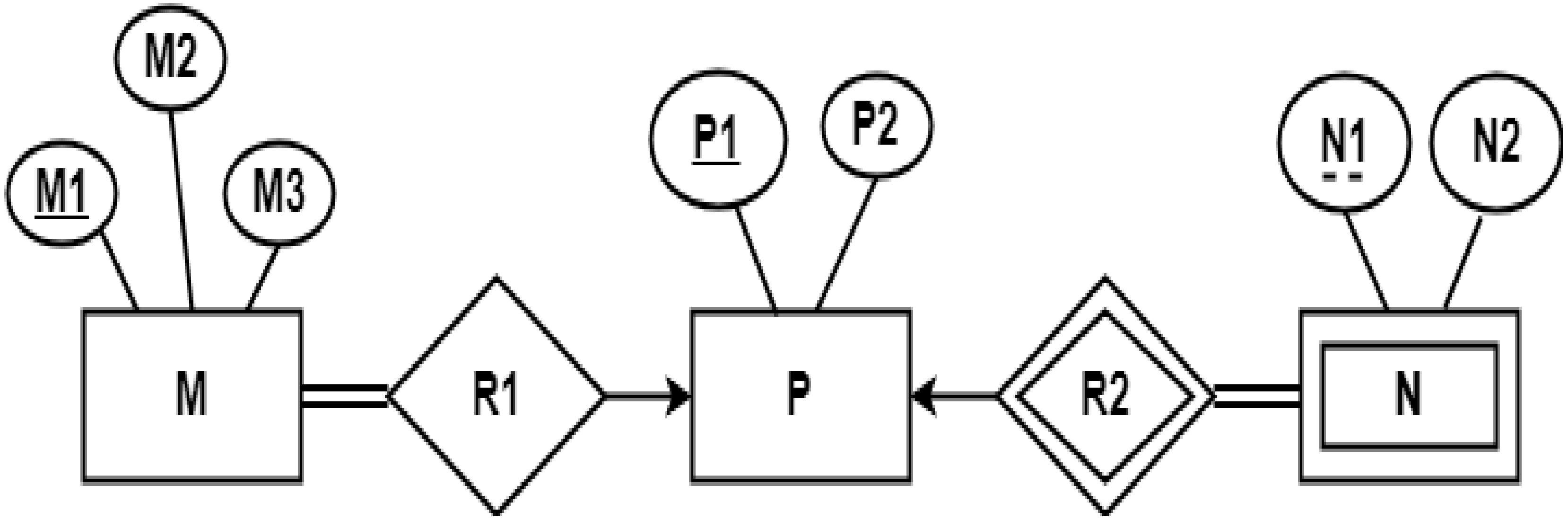
A (<u>a1</u> , a2)

BR (<u>a1</u> , <u>b1</u> , b2)



Problem-01

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



Problem-01

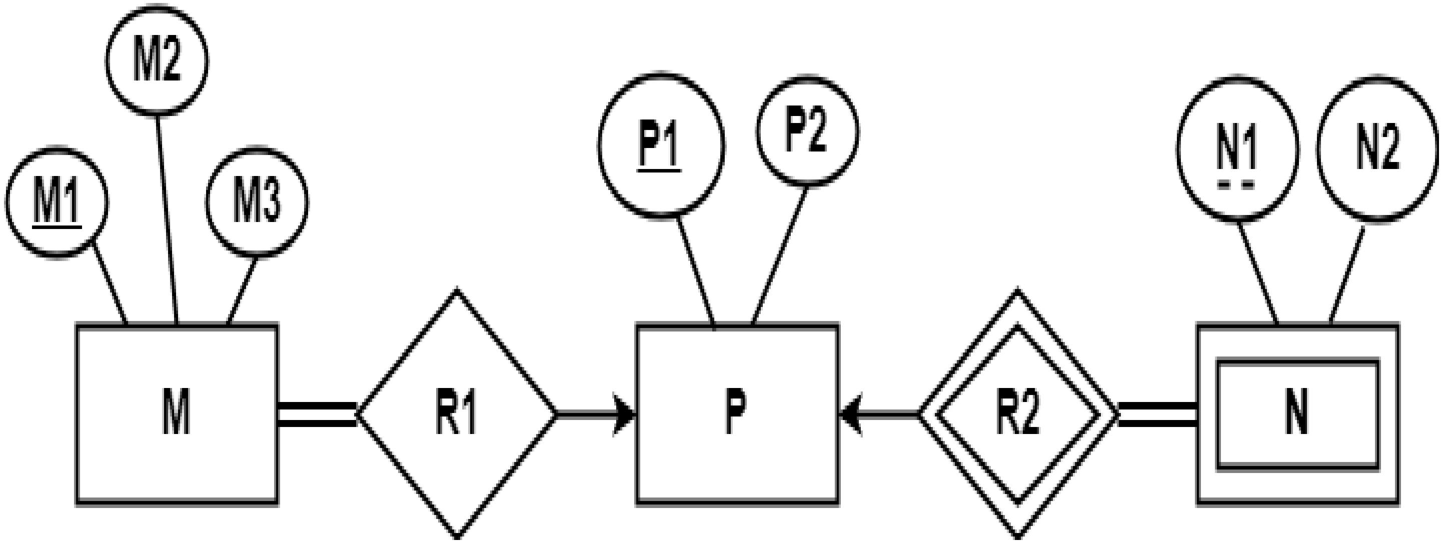
Solution

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

MR1 (<u>M1</u> , M2 , M3 , P1)

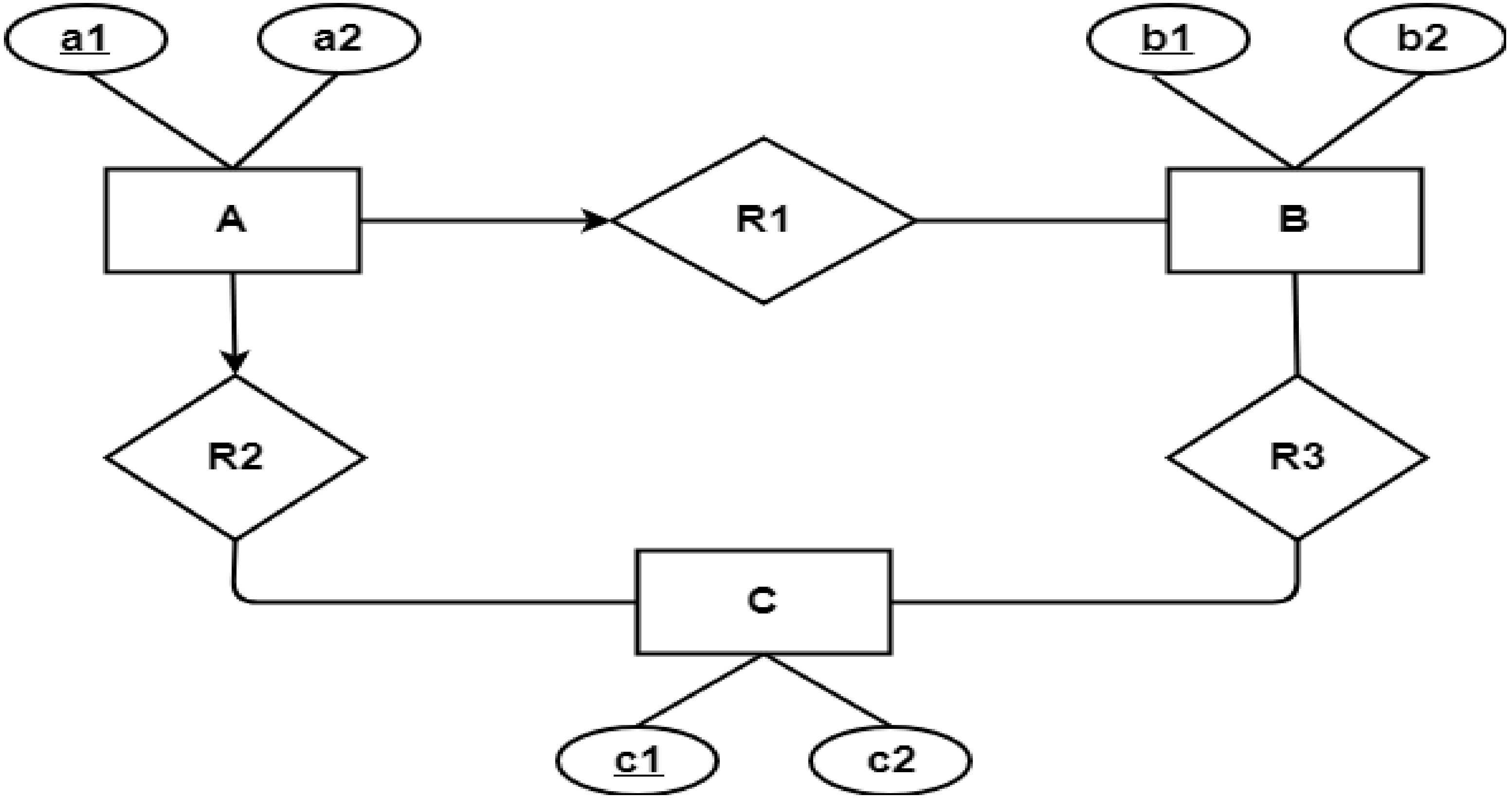
P (<u>P1</u> , P2)

NR2 (<u>P1</u> , <u>N1</u> , N2)



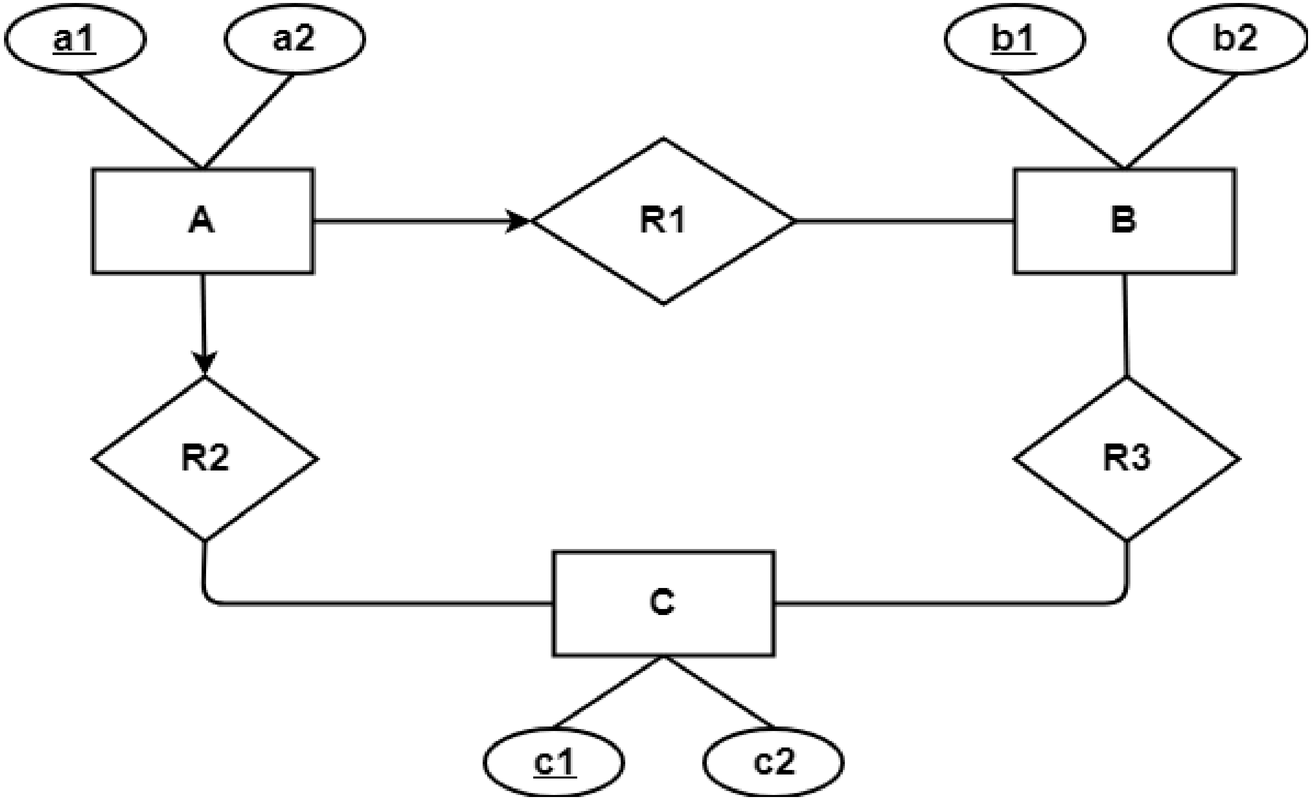
Problem-02

- Find the minimum number of tables required to represent the given ER diagram in relational model



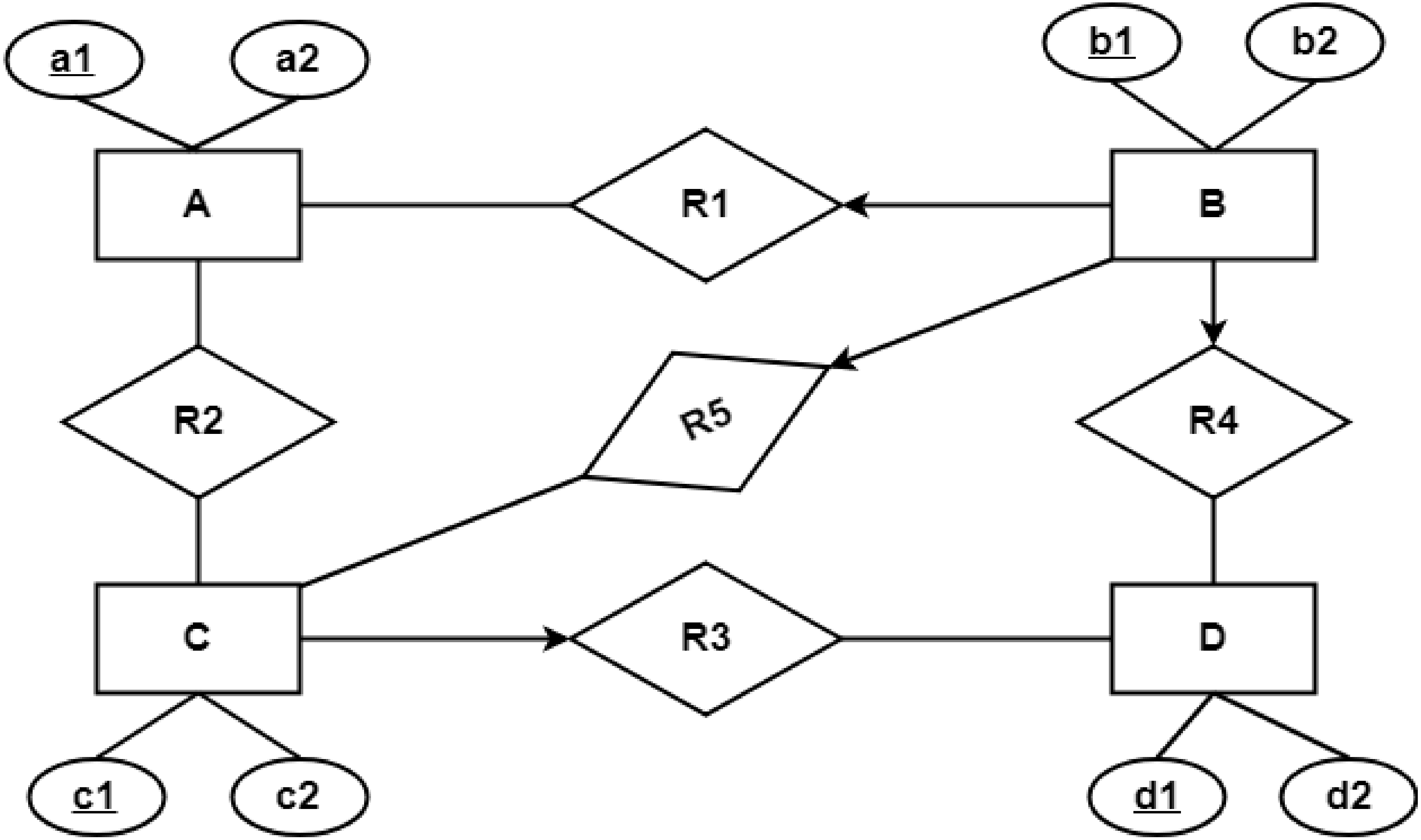
Solution

- Applying the rules, minimum 4 tables will be required-
  - AR1R2 (a1 , a2 , b1 , c1)
  - B (b1 , b2)
  - C (c1 , c2)
  - R3 (b1 , c1)



GLA UNIVERSITY MATHURA
Recognised by UGC Under Section 2(f)
Accredited with A Grade by NAAC
12-B Status from UGC

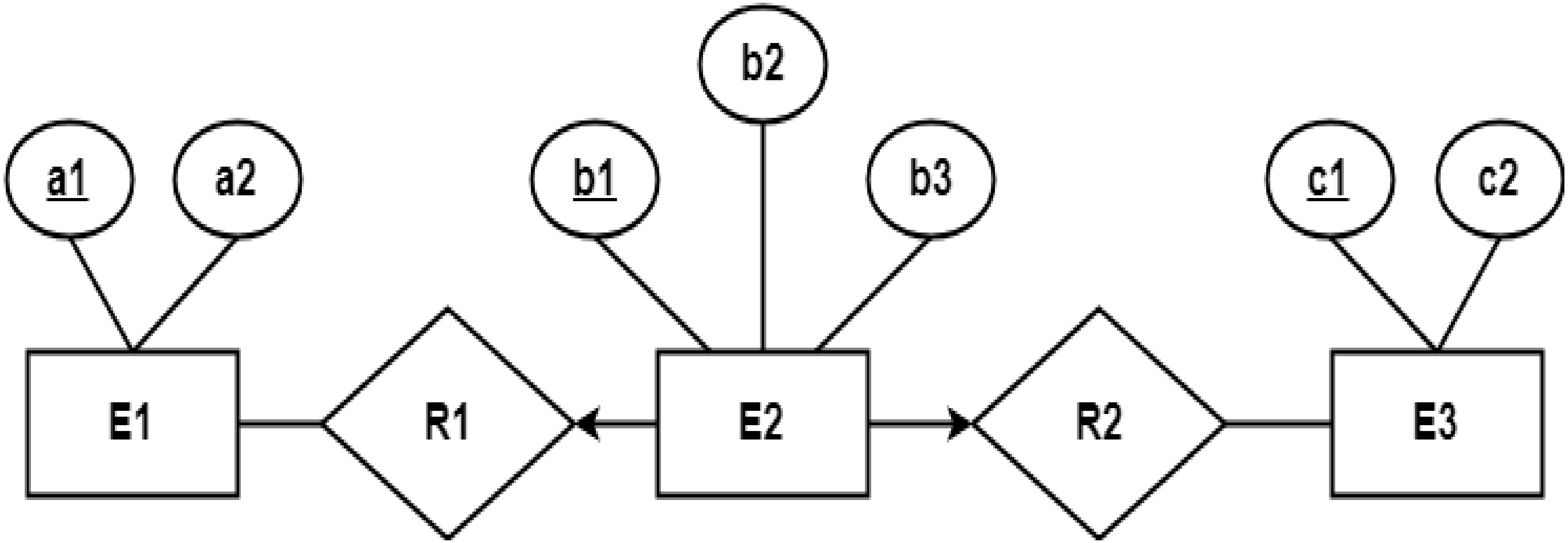
Problem-03



Solution

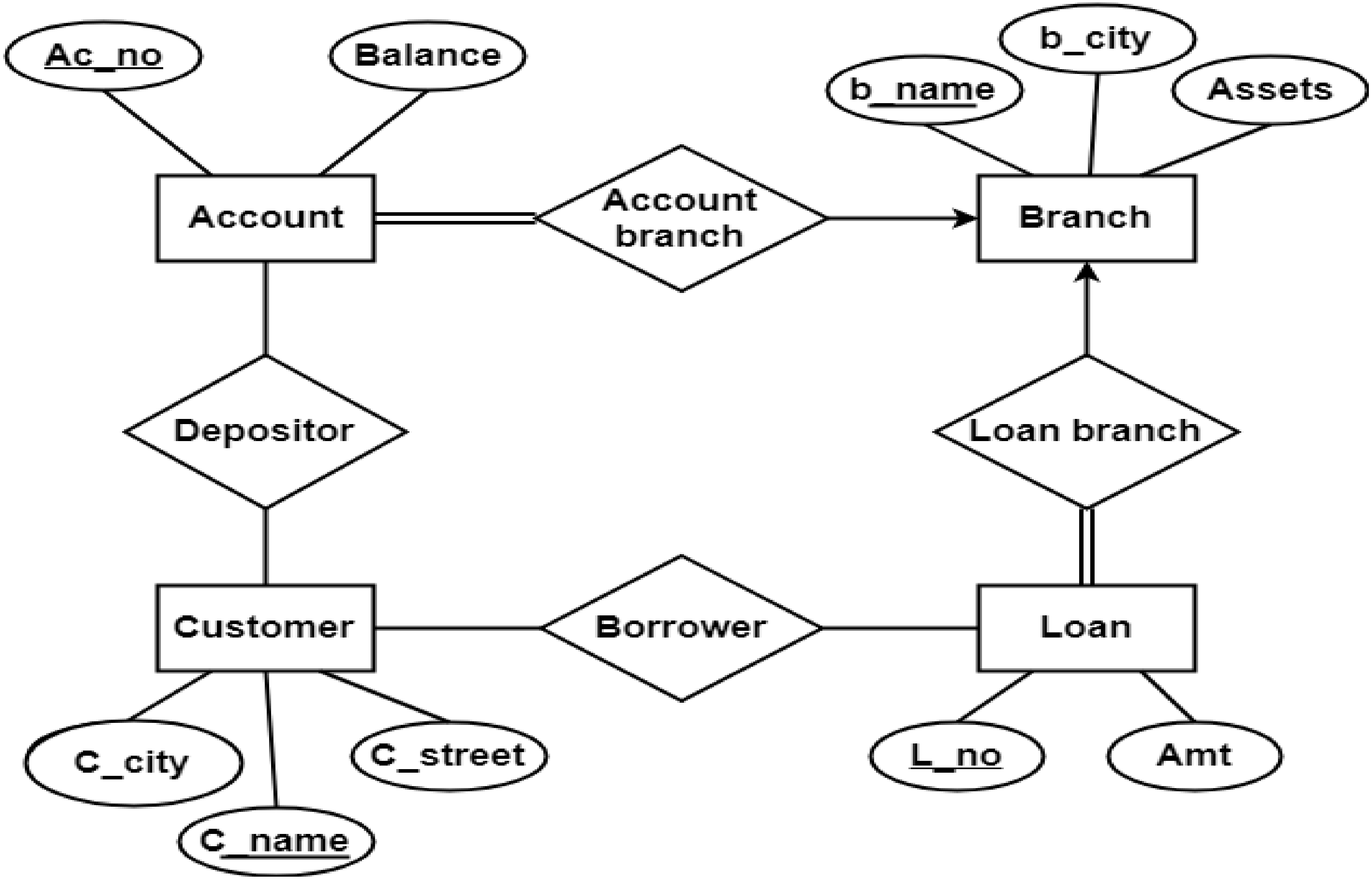
- BR1R4R5 (b1 , b2 , a1 , c1 , d1)
- A (a1 , a2)
- R2 (a1 , c1)
- CR3 (c1 , c2 , d1)
- D (d1 , d2)

Problem-04



- Applying the rules, minimum 3 tables will be required
- E1 (a1 , a2)
- E2R1R2 (b1 , b2 , a1 , c1 , b3)
- E3 (c1 , c2)

Problem-05


Ac_no
Balance
b_city
b_name
Assets
Account
Account branch
Branch
Depositor
Loan branch
Customer
Borrower
Loan
C_city
C_street
C_name
L_no
Amt

Solution

- Applying the rules that we have learnt, minimum 6 tables will be required-

  Account (Ac_no , Balance , b_name)

  Branch (b_name , b_city , Assets)

  Loan (L_no , Amt , b_name)

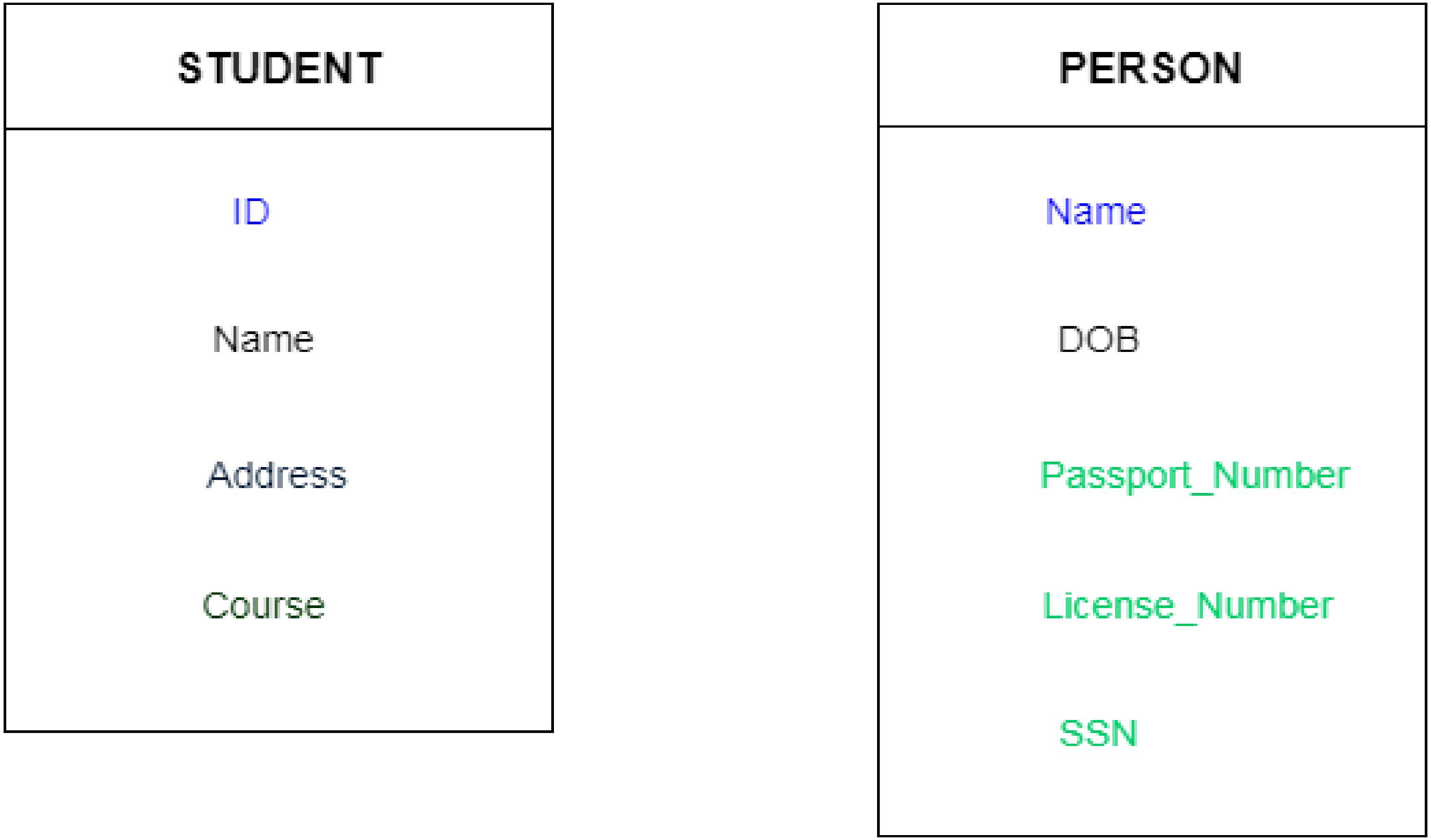
  Borrower (C_name , L_no)

  Customer (C_name , C_street , C_city)

  Depositor (C_name , Ac_no)

Keys

- Keys play an important role in the relational database.
- It is used to uniquely identify any record or row of data from the table. It is also used to establish and identify relationships between tables.
- **For example:** In Student table, ID is used as a key because it is unique for each student. In PERSON table, passport_number, license_number, SSN are keys since they are unique for each person.



STUDENT

ID

Name

Address

Course

PERSON

Name

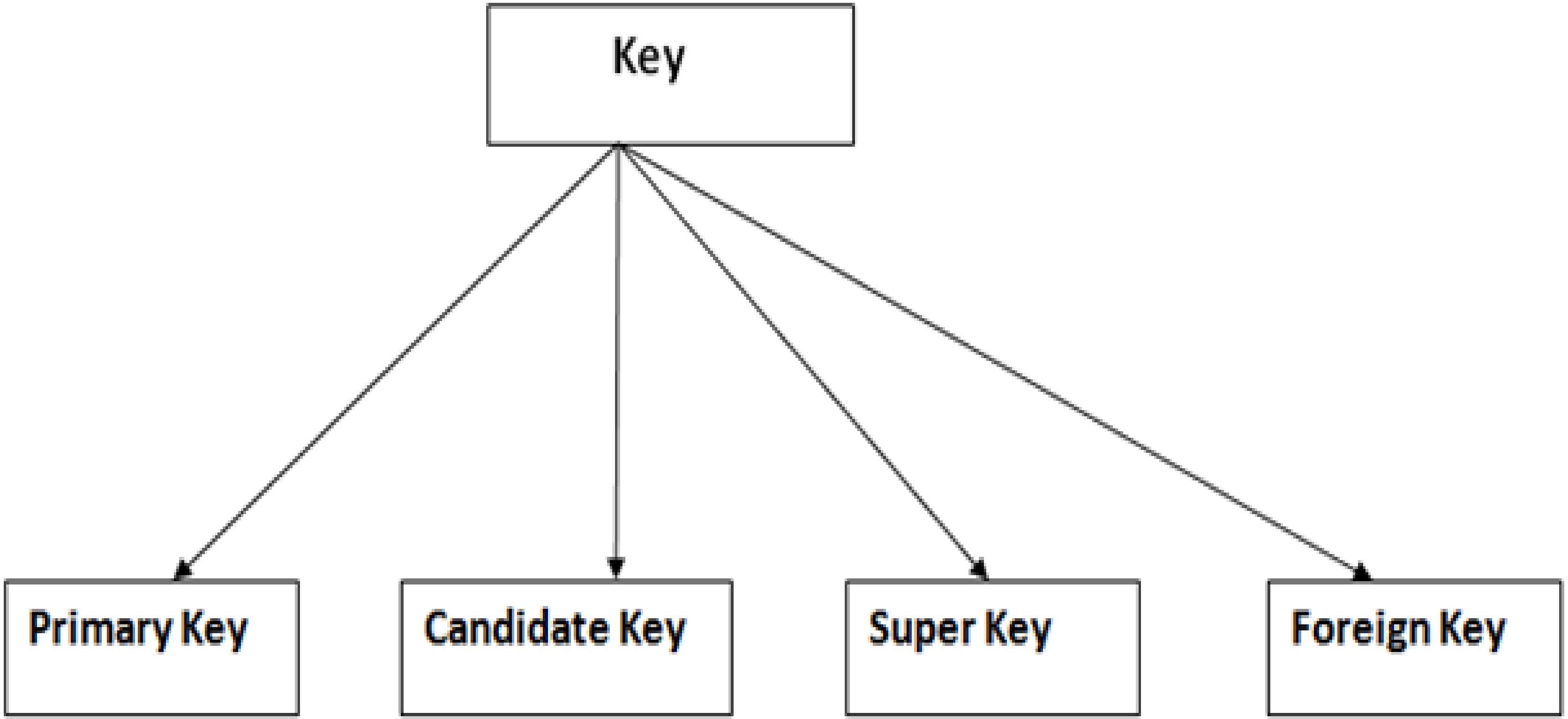
DOB

Passport_Number

License_Number

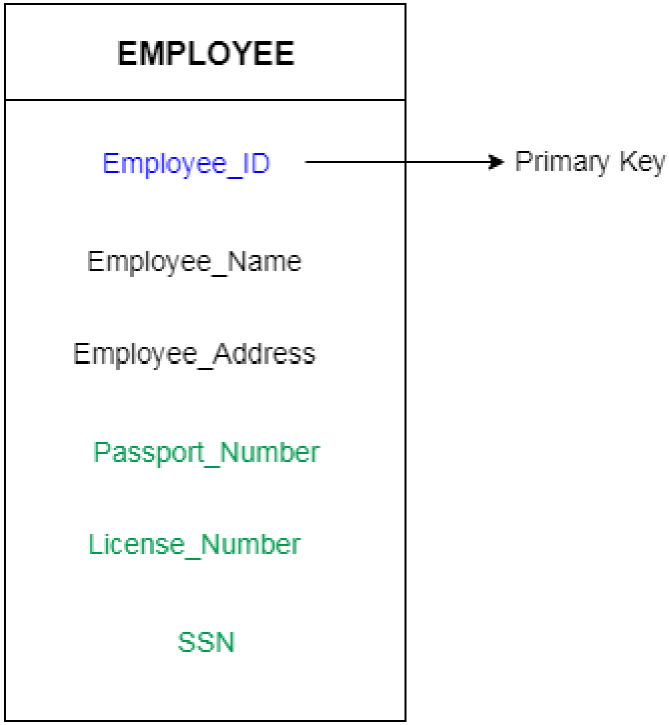
SSN

Types of key



1. Primary key

- It is the first key which is used to identify one and only one instance of an entity uniquely.
- An entity can contain multiple keys as we saw in PERSON table.
- The key which is most suitable from those lists become a primary key.
- In the EMPLOYEE table, ID can be primary key since it is unique for each employee. In the EMPLOYEE table, we can even select License_Number and Passport_Number as primary key since they are also unique.
- For each entity, selection of the primary key is based on requirement and developers.



EMPLOYEE

Employee_ID → Primary Key

Employee_Name

Employee_Address

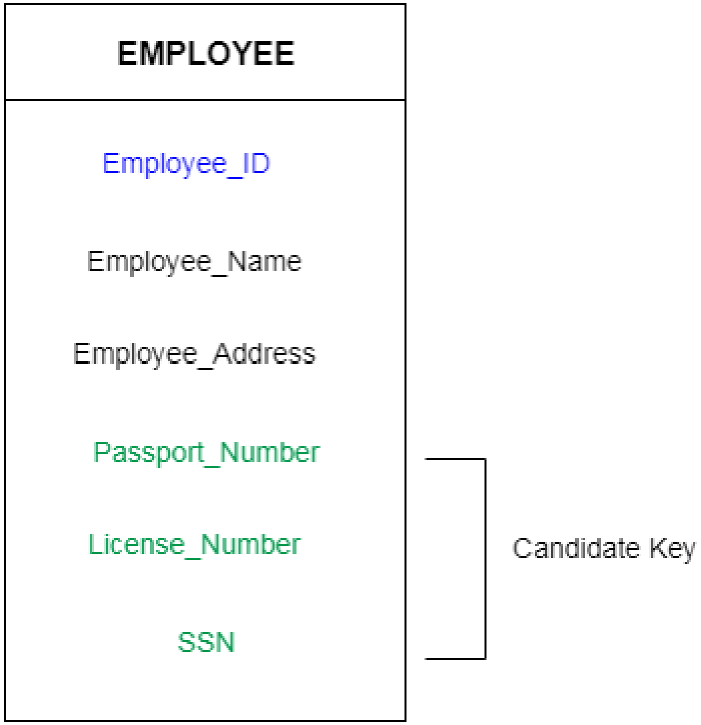
Passport_Number

License_Number

SSN

2. Candidate key

- A candidate key is an attribute or set of an attribute which can uniquely identify a tuple.
- The remaining attributes except for primary key are considered as a candidate key.
- The candidate keys are as strong as the primary key.
- **For example:** In the EMPLOYEE table, id is best suited for the primary key. Rest of the attributes like SSN, Passport_Number, and License_Number, etc. are considered as a candidate key.



EMPLOYEE

Employee_ID

Employee_Name

Employee_Address

Passport_Number

License_Number

SSN

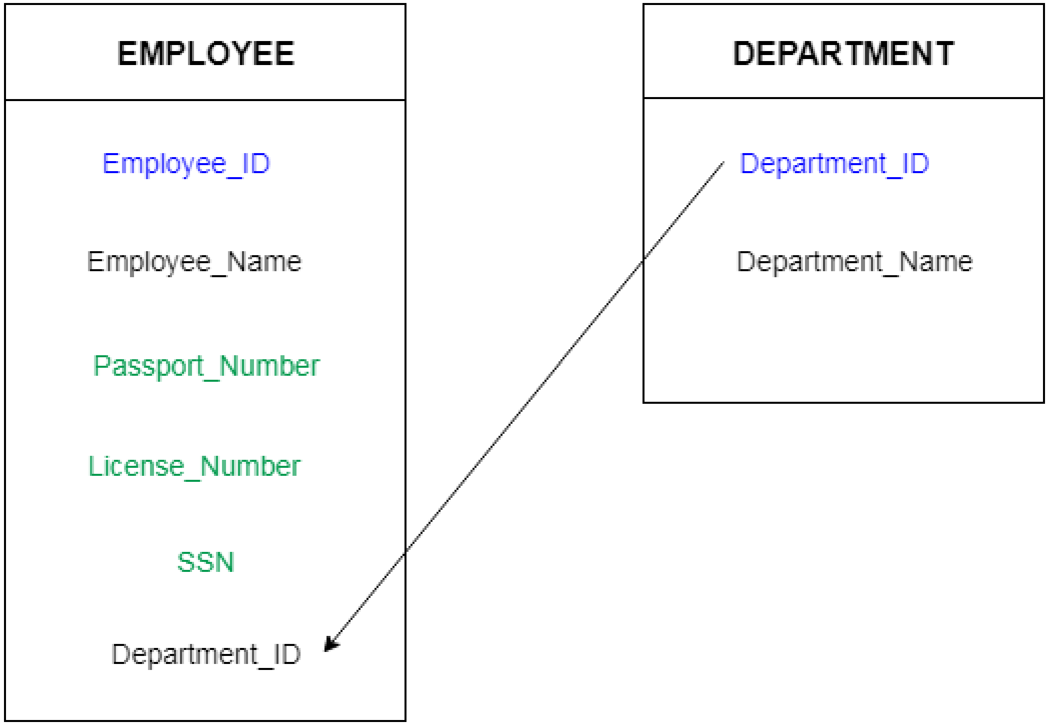
Candidate Key

3. Super Key

- Super key is a set of an attribute which can uniquely identify a tuple. Super key is a superset of a candidate key.
- **For example:** In the above EMPLOYEE table, for(EMPLOEE_ID, EMPLOYEE_NAME) the name of two employees can be the same, but their EMPLYEE_ID can't be the same. Hence, this combination can also be a key.
- The super key would be EMPLOYEE-ID, (EMPLOYEE_ID, EMPLOYEE-NAME), etc.

4. Foreign key

- Foreign keys are the column of the table which is used to point to the primary key of another table.
- In a company, every employee works in a specific department, and employee and department are two different entities.
- So we can't store the information of the department in the employee table.
- That's why we link these two tables through the primary key of one table.
- We add the primary key of the DEPARTMENT table, Department_Id as a new attribute in the EMPLOYEE table.
- Now in the EMPLOYEE table, Department_Id is the foreign key, and both the tables are related.



EMPLOYEE

Employee_ID

Employee_Name

Passport_Number

License_Number

SSN

Department_ID

DEPARTMENT

Department_ID

Department_Name