Database Management System (BIT-17)

B. Tech-IVth Sem (2020-21)

Dr. D. S. Singh
Associate Professor
Department of ITCA
MMMUT Gorakhpur
Email: dssitca@mmmut.ac.in

Database Management System

Course Objectives

- know and understand the fundamental concepts, techniques, and terminology of the database management system.
- know and understand the basic syntax, semantics, and pragmatics of SQL & PL/SQL.
- can analyze problems and apply DBMS concepts and techniques to develop appropriate programs to solve the problems,

Database Management System

- can evaluate alternative database designs to determine which are better according to selected criteria.
- know and understand the basic features of database transactions and concurrency control.
- are able to reason about and manipulate concurrency control techniques.

Database Management System

Learning Outcomes

- list and define the fundamental concepts of DBMS.
- manually execute a simple database design a transaction over it.
- manually infer the type of a given database transaction.
- implement algorithms and data structures as database transaction.
- design databases that are modular and have reusable components.
- explain concurrency control and concurrent transactions.

Unit-I

Introduction

Introduction: An Overview and motivation of Database Management System, Characteristics of database approach, Advantages of using DBMS approach, Database System vs File System, Database System Concept and Architecture, Data Model & its types, Schema and Instances, Data Independence, Database Language and Interfaces, Data Definitions Language, DML, Overall Database Structure.

Data Modelling using Entity Relationship Model: E-R Model Concepts, Notation for E-R Diagram, Mapping Constraints, Keys, Concepts of Super Key, Candidate Key, Primary Key, Generalization, Aggregation, Reduction of An ER Diagrams to Tables, Extended ER Model, Relationship of Higher Degree.

Unit-II

Relational data Model and Language

Relational Data Model Concepts, Consistency constraints, Integrity Constraints, Entity Integrity, Referential Integrity, Keys Constraints, Domain Constraints, Relational Algebra, Relational Calculus, Tuple and Domain Calculus.

Introduction to SQL:

Characteristics of SQL, Advantage of SQL, SQL Data Type and Literals, Types of SQL Commands, SQL Operators and their Procedure, Tables, Views, and Indexes, Queries and Sub Queries, Aggregate Functions, Insert, Update and Delete Operations, Joins, Unions, Intersection, Minus, Cursors, Triggers, Procedures in SQL/PL SQL

Unit-III

Data Base Design & Normalization

Database Anomalies, Functional Dependencies, Normal Forms, First, Second, Third Normal Forms, BCNF, Fourth and Fifth Normal Forms, Closure of a set of FDs and MVDs, Armstrong's axioms, Minimal or Canonical cover of FDs, Inclusion Dependence, Loss Less Join Decompositions, Dependency Preserving, Normalization using FD, MVD, and JDS, Alternative Approaches to Database Design.

Unit-IV

Transaction Processing Concepts

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

Distributed Database

Characteristics of Distributed Database, Advantages and disadvantages of distributed database, Distributed Data Storage.

Concurrency Control Techniques

Concurrency Control, Locking Techniques for Concurrency Control, Time Stamping Protocols for Concurrency Control, Validation Based Protocol, Multiple Granularity, Multi Version Schemes, Recovery with Concurrent Transaction.

Books & References

- Date C J, “An Introduction To Database System”, Addision Wesley.
- Korth, Silberchatz, Sudarshan, “Database Concepts”, McGraw Hill.
- Elmasri, Navathe, “Fundamentals Of Database Systems”, Addision Wesley.
- Leon & Leon, “Database Management System”, Vikas Publishing House
- Bipin C. Desai, “An introduction to Database Systems”, Galgotia Publication.
- Majumdar & Bhattacharya, “Database Management System”, TMH.
- Ramakrishnan, Gehrke, “Database Management System”, McGraw Hill.
- Kroenke, “Database Processing: Fundamentals, Design and Implementation”, Pearson Education.
- Maheshwari Jain, “DBMS: Complete Practical Approach”, Firewall Media, New Delhi.

Overview of Database Management System

- **Data** is a collection of facts and figures that can be processed to produce the information.
- Processed data is known as **information.**
- **Database** is a collection of inter-related data.
- Database Management System = Database + A set of program to access the data/information from database.
- A **DBMS** stores data in such a way that it becomes easier to retrieve, manipulate, and produce information.

Characteristics of DBMS

- **Real-world entity**
- **Relation-based tables**
- **Isolation of data and application**
- **Less redundancy**
- **Consistency**
- **Query Language**
- **ACID Properties:** ***Atomicity, Consistency, Isolation and Durability***
- **Multiuser and Concurrent Access**
- **Multiple views**
- **Security**

Users of DBMS

- **Administrator**
- **Designers**
- **End users**

Advantages of DBMS

- **Controls database redundancy:** It can control data redundancy because it stores all the data in one single database file and that recorded data is placed in the database.

- **Data sharing:** In DBMS, the authorized users of an organization can share the data among multiple users.

- **Easily Maintenance:** It can be easily maintainable due to the centralized nature of the database system.

Advantages of DBMS

- **Reduce time:** It reduces development time and maintenance need.

- **Backup:** It provides backup and recovery subsystems which create automatic backup of data from hardware and software failures and restores the data if required.

- **Multiple User Interface:** It provides different types of user interfaces like graphical user interfaces, application program interfaces.

Disadvantages of DBMS

- **Cost of Hardware and Software:** It requires a high speed of data processor and large memory size to run DBMS software.

- **Size:** It occupies a large space of disks and large memory to run them efficiently.

- **Complexity:** Database system creates additional complexity and requirements.

- **Higher impact of failure:** Failure is highly impacted the database because in most of the organization, all the data stored in a single database and if the database is damaged due to power failure or database corruption then the data may be lost forever.

Database Management Vs File Management System

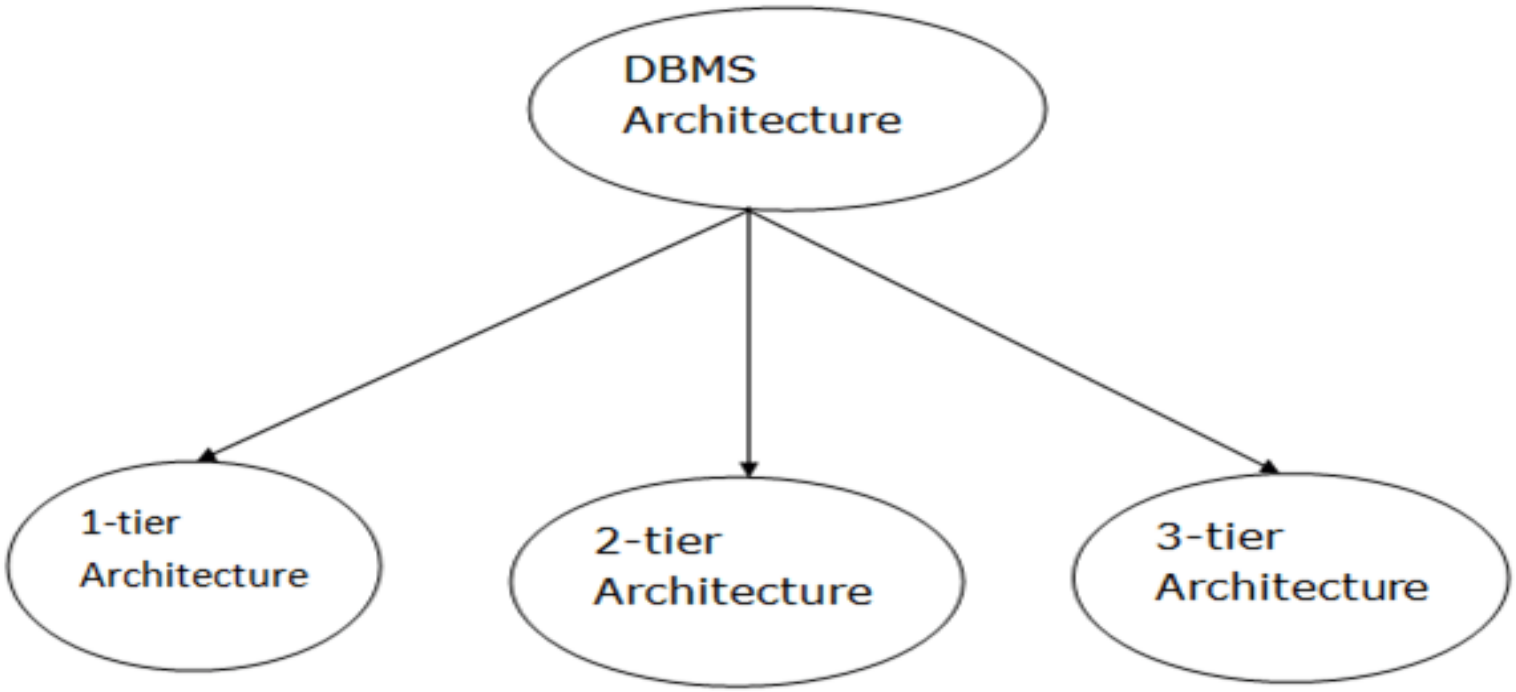
File Management System	Database Management System
Small systems like C++ or Java program.	Large systems like Oracle or Sybase.
Relatively cheap	Relatively expensive
Have simple structure	Have complex structure
Needs very little preliminary design.	Needs vast preliminary design
Not secure	More secure
Often single user- oriented.	Multiple user-oriented.
Have isolated data and simple backup/recovery mechanism	Have shared data and complex & sophisticated backup/recovery mechanism.

Database System Concepts and Architecture

DBMS Architecture

- The DBMS design depends upon its architecture. The basic client/server architecture is used to deal with a large number of PCs, web servers, database servers and other components that are connected with networks.
- The client/server architecture consists of many PCs and a workstation which are connected via the network.
- DBMS architecture depends upon how users are connected to the database to get their request done.

Types of DBMS Architecture



1-Tier Architecture

- In this architecture, the database is directly available to the user. It means the user can directly have the DBMS and use it.

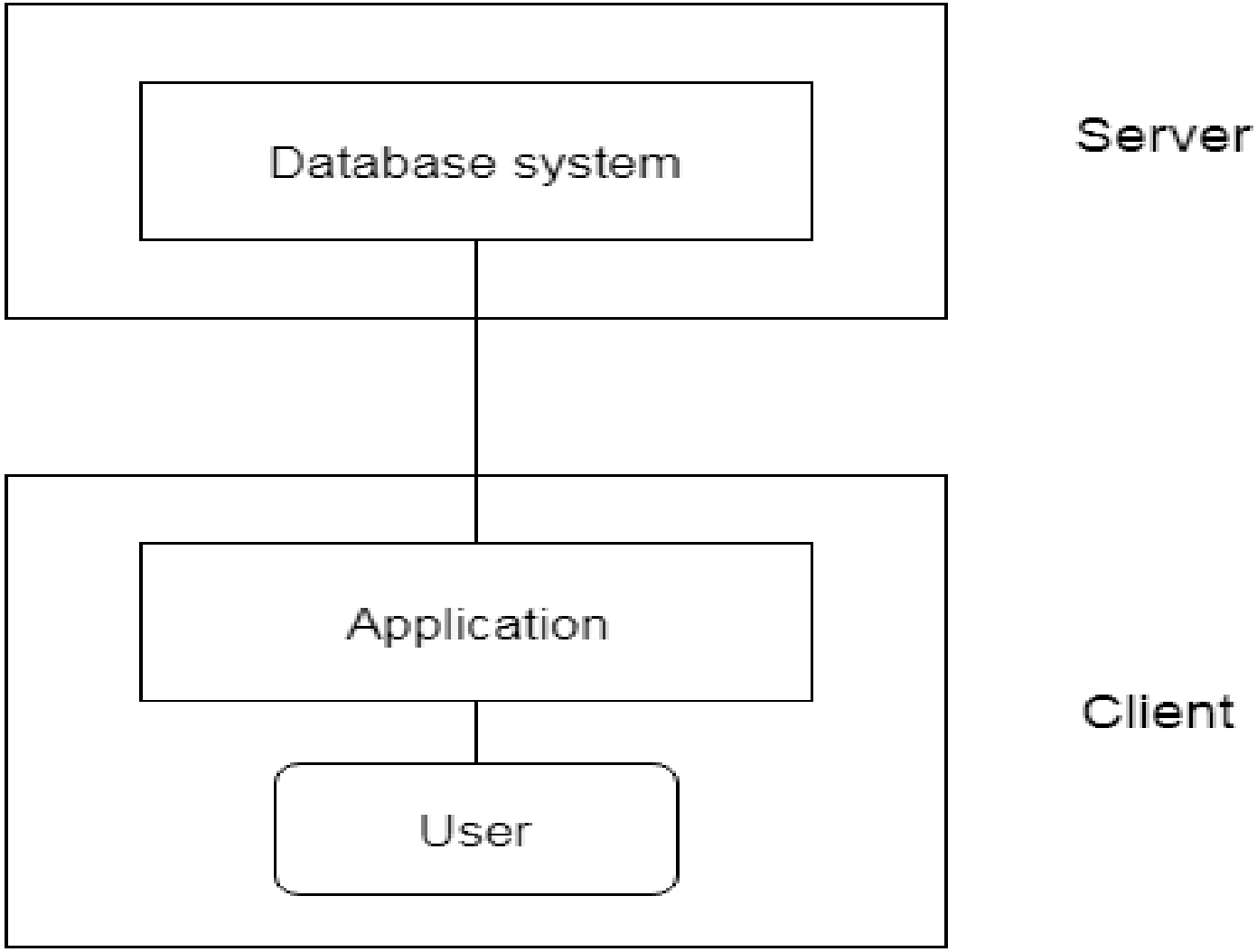
- Any changes done here will directly be done on the database itself. It doesn't provide a handy tool for end users.

- 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

- The 2-Tier architecture is same as basic client-server. In the two-tier architecture, applications on the client end can directly communicate with the database at the server side. For this interaction, API's like: ODBC, JDBC are used.
- The user interfaces and application programs are run on the client-side.
- The server side is responsible to provide the functionalities like: query processing and transaction management.
- To communicate with the DBMS, client-side application establishes a connection with the server side.

2-Tier Architecture

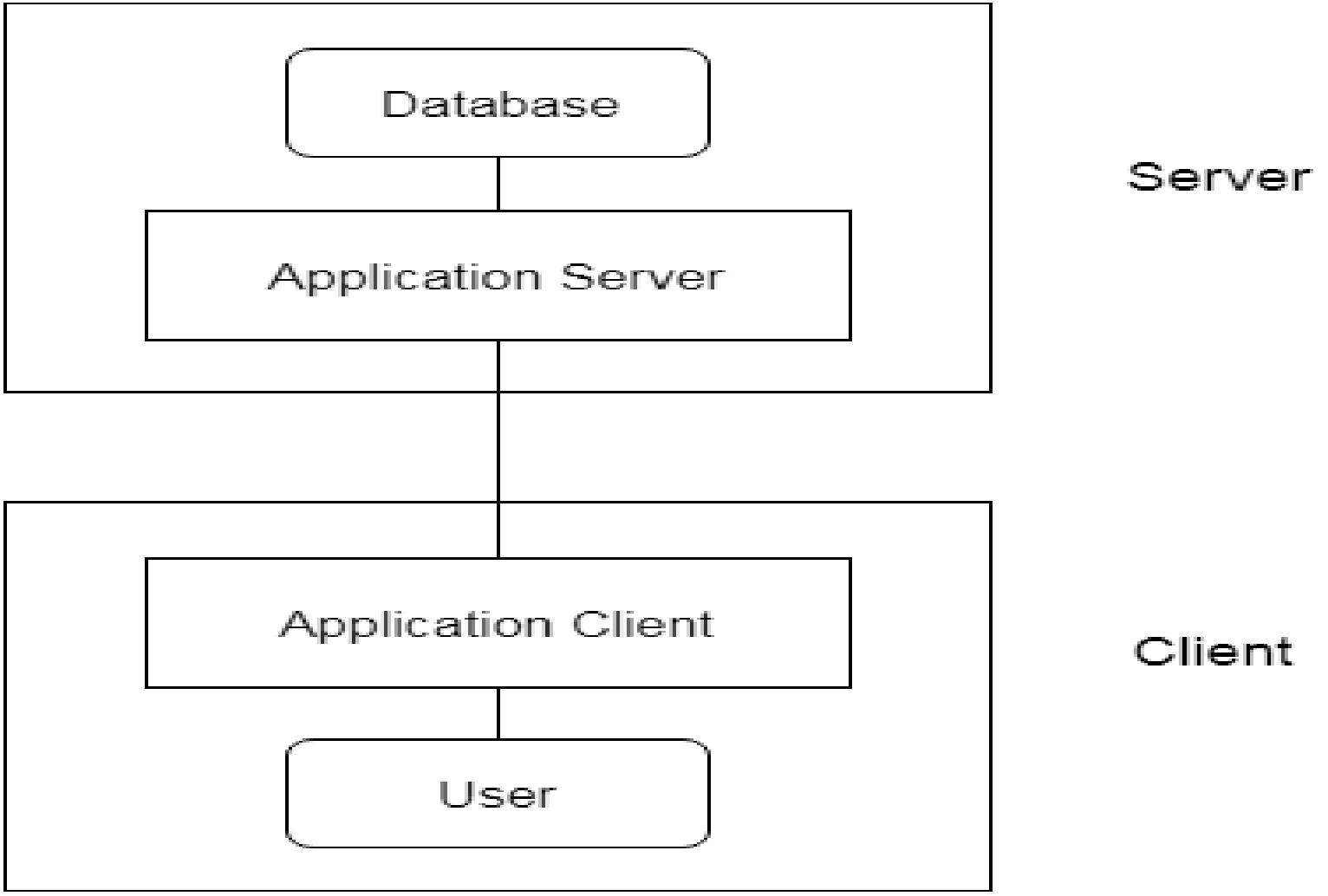


3-Tier Architecture

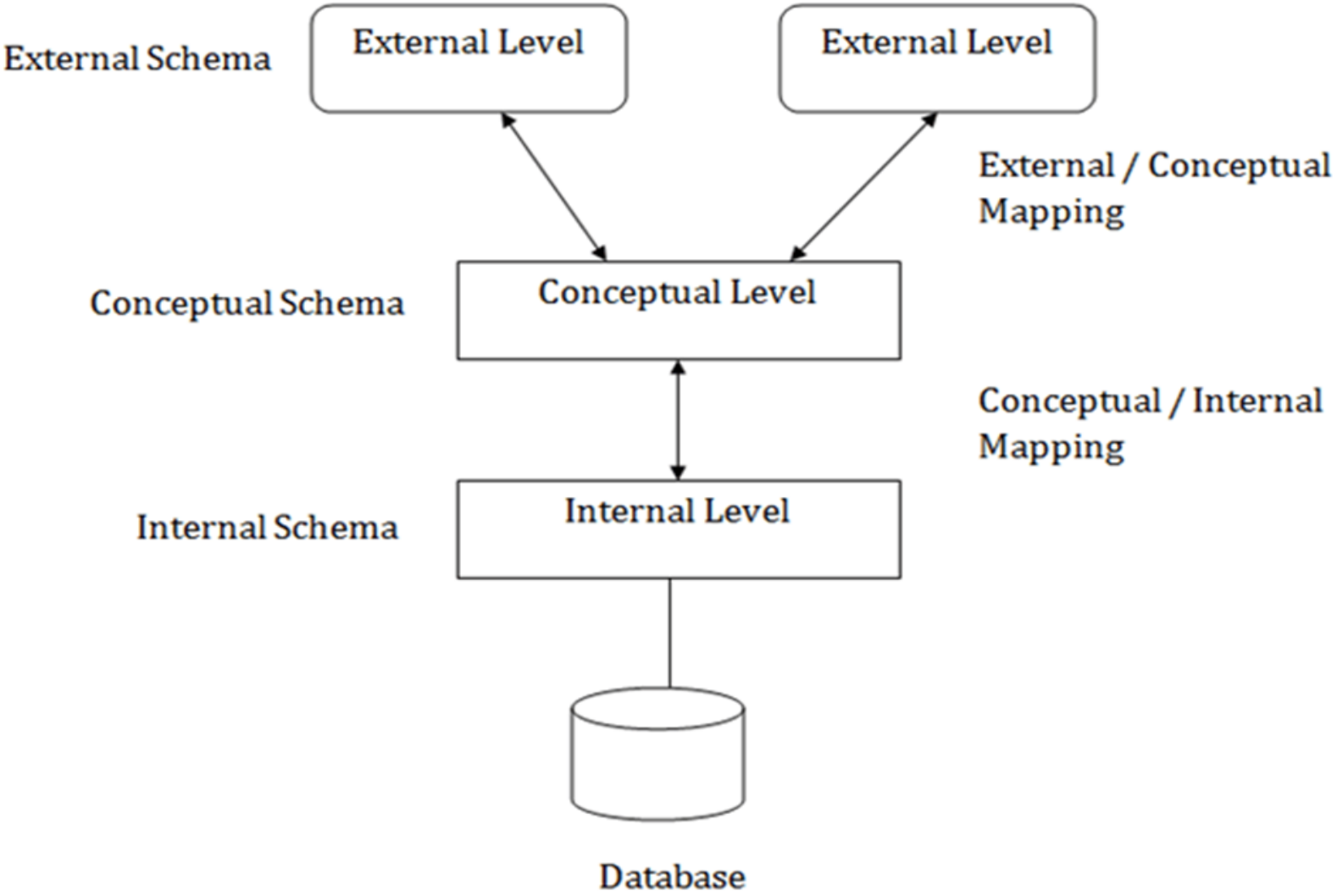
- The 3-Tier architecture contains another layer between the client and server. In this architecture, client can't directly communicate with the server.
- The application on the client-end interacts with an application server which further communicates with the database system.
- End user has no idea about the existence of the database beyond the application server. The database also has no idea about any other user beyond the application.

- The 3-Tier architecture is used in case of large web

3-Tier Architecture



Three Level Architecture of DBMS



Three Level Architecture of DBMS

- It is also known as three schema architecture of DBMS.
- Mapping is used to transform the request and response between various database levels of architecture.
- Mapping is not good for small DBMS because it takes more time.
- In External/Conceptual mapping, it is necessary to transform the request from external level to conceptual schema.
- In Conceptual/Internal mapping, DBMS transform the request from the conceptual to internal level.

Internal Level

- The internal level has an internal schema which describes the physical storage structure of the database.
- The internal schema is also known as a physical schema.
- It uses the physical data model. It is used to define that how the data will be stored on the disk.
- Physical level is used to describe complex low-level data structures in detail.

Conceptual Level

- It describes the design of a database at the conceptual level. Conceptual level is also known as logical level.
- It also describes the structure of the whole database.
- It also describes what data are to be stored in the database and what relationship exists among those data.
- In the conceptual level, internal details such as an implementation of the data structure are hidden.
- Programmers and database administrators work at this level.

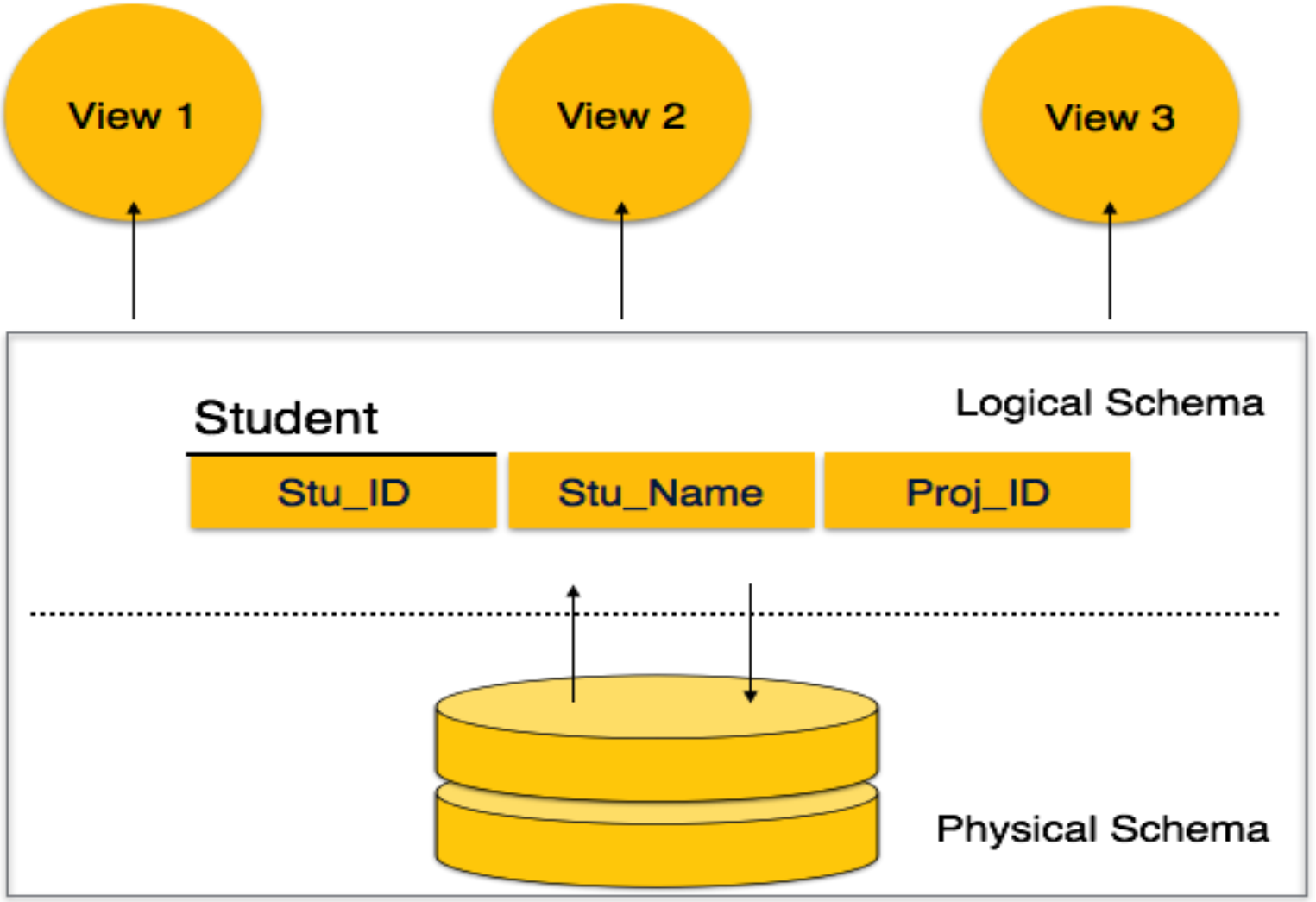
External Level

- At the external level, a database contains several schemas that sometimes called as subschema. The subschema is used to describe the different view of the database.
- An external schema is also known as view schema.
- Each view schema describes the database part that a particular user group is interested and hides the remaining database from that user group.
- The view schema describes the end user interaction with database systems.

Database Schema

- Skeleton that represents the logical view of the entire database.
- Defines how the data is organized and how the relations among them are associated.
- Formulates all the constraints that are to be applied on the data.
- Defines its entities and the relationship among them.
- Contains a descriptive detail of the database, which can be depicted by means of schema diagrams.
- It's the database designers who design the schema to help programmers to understand the database and make it useful.

Database Schema



Database Schema

A database schema can be divided broadly into two categories:

- Physical Database Schema: This schema pertains to the actual storage of data and its form of storage like files, indices, etc. It defines how the data will be stored in a secondary storage.
- Logical Database Schema: This schema defines all the logical constraints that need to be applied on the data stored. It defines tables, views, and integrity constraints.

Database Instance

- Database schema is the skeleton structure of database.
- It is designed when the database doesn't exist at all.
- Once the database is operational, it is very difficult to make any changes to it.
- A database schema does not contain any data or information.
- A database instance is a state of operational database with data at any given time.

Database Instance

- A DBMS ensures that its every instance (state) is in a valid state, by diligently following all the validations, constraints, and conditions that the database designers have imposed.
- If a database system is not multi-layered, then it becomes difficult to make any changes in the database system. Database systems are designed in multi-layers.

The collection of information stored in the database at a particular moment is called as an instance of the database.

Data Models

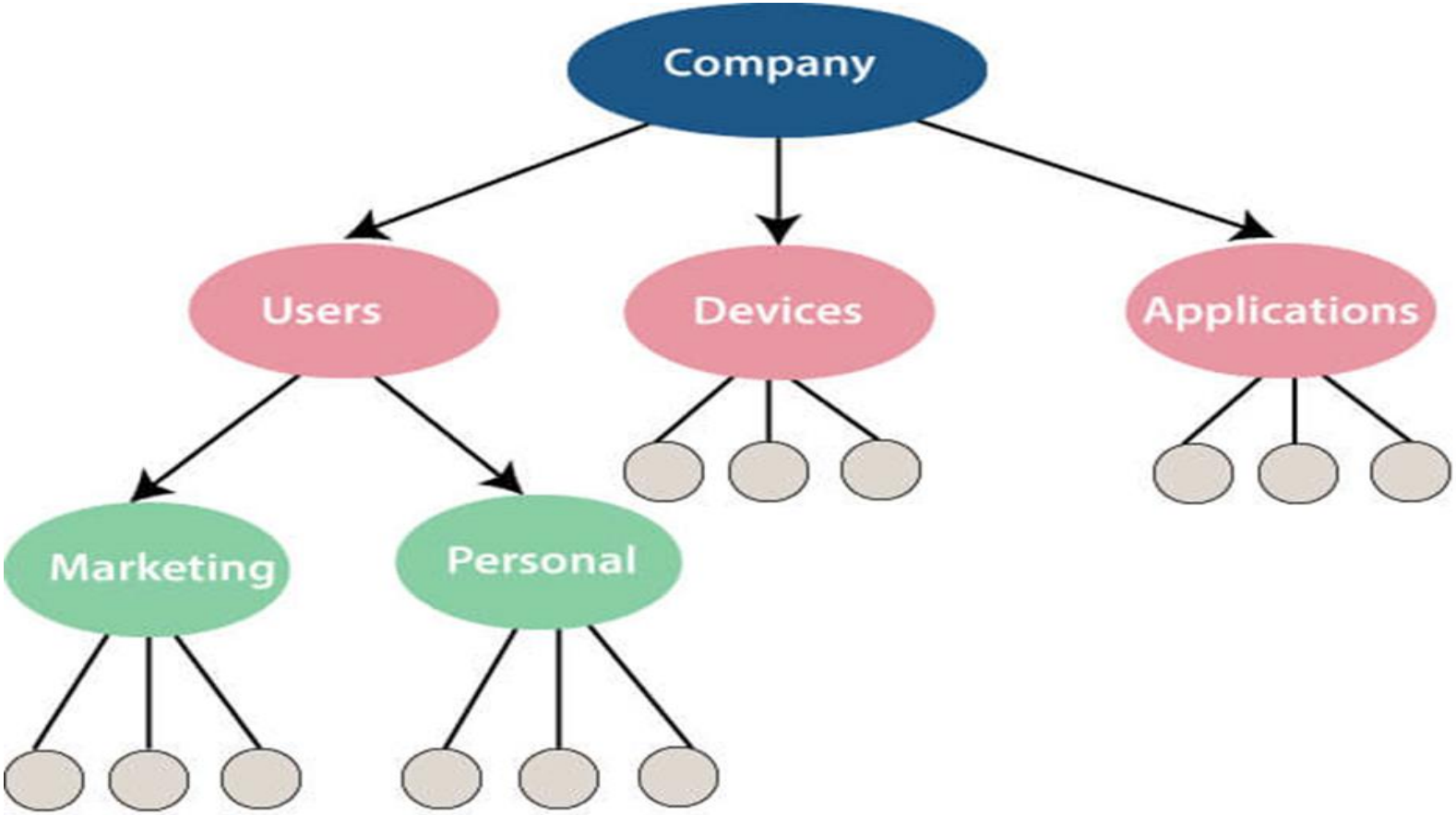
- Data models define how the logical structure of a database is modeled.
- Data Models are fundamental entities to introduce abstraction in a DBMS.
- Data models define how data is connected to each other and how they are processed and stored inside the system.
- The very first data model could be flat data-models, where all the data used are to be kept in the same plane.
- Earlier data models were not so scientific; hence they were prone to introduce lots of duplication and update anomalies.

Hierarchical Data Model

- In this model, files are related in a parent/child manner.
- Like file system, this model also had some limitations like complex implementation, lack structural independence.
- It can't easily handle a many-many relationship, etc.
- It is also called IMS (Information Management System).
- Below diagram represents Hierarchical Data Model. Small circle represents objects.

Hierarchical Data Model

Example of HDM



Network Data Model

- In this model, files are related as owners and members.

- **Network data model identified the following components:**

  - Network schema (Database organization)
  - Sub-schema (views of database per user)
  - Data management language (procedural)

- This model also had some limitations like system complexity and difficult to design and maintain.

Relational Data Model

- Relational Data Model has two main terminologies called instance and schema.
- The instance is a table with rows or columns.
- Schema specifies the structure like name of the relation, type of each column and name.
- This model uses some mathematical concept like set theory and predicate logic.
- During the era of the relational database, many models introduced such as object-oriented model, object-relational model, etc.

Data Independence

- A database system normally contains a lot of data in addition to users' data. For example, it stores data about data, known as metadata, to locate and retrieve data easily.
- It is rather difficult to modify or update a set of metadata once it is stored in the database.
- But as a DBMS expands, it needs to change over time to satisfy the requirements of the users.
- If the entire data is dependent, it would become a tedious and highly complex job.

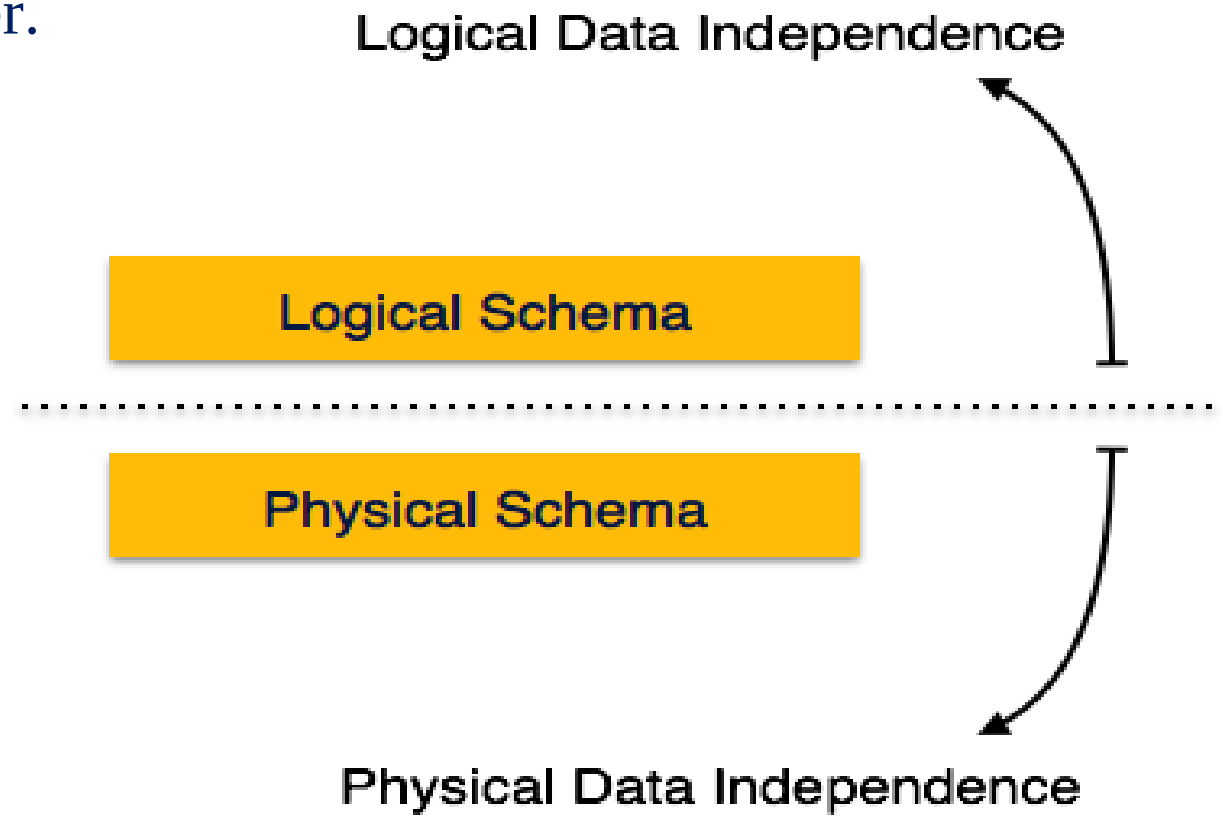
Data Independence

- The major objective of 3-level architecture of DBMS is to provide data independence which means that upper levels are unaffected by changes in lower levels.

- There are two types of data independence:
  - ✓ Physical Data Independence
  - ✓ Logical Data Independence

Data Independence

- Metadata itself follows a layered architecture, so that when we change data at one layer, it does not affect the data at another level. This data is independent but mapped to each other.



Physical Data Independence

- It indicates that the physical storage structures or devices could be changed without affecting the conceptual schema.
- All the schemas are logical, and the actual data is stored in bit format on the disk.
- Physical data independence is the power to change the physical data without impacting the schema or logical data.
- For example, in case we want to change or upgrade the storage system itself − suppose we want to replace hard-disks with SSD − it should not have any impact on the logical data or schemas.

Logical Data Independence

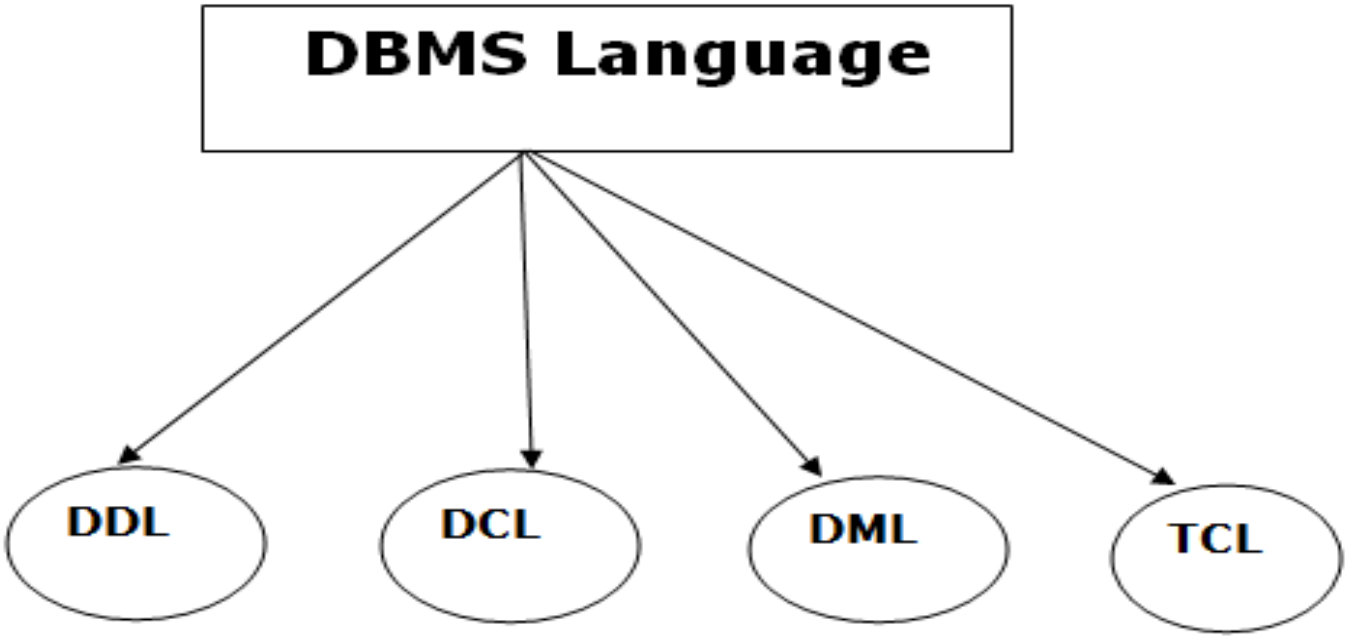
- Logical data independence means that the conceptual schema (middle level) can be changed without affecting the existing external schemas.

- 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.

Database Language and Interfaces

- A DBMS has appropriate languages and interfaces to express database queries and updates.
- Database languages can be used to read, store and update the data in the database. It can be classified as:



Data Definition Language (DDL)

- **DDL** stands for **D**ata **D**efinition **L**anguage. It is used to define database structure or pattern.
- It is used to create schema, tables, indices, constraints, etc. in the database.
- Using the DDL statements, we can create the skeleton of the database.
- Data definition language is used to store the information of metadata like the number of tables and schemas, their names, indexes, columns in each table, constraints, etc.

Data Definition Language (DDL)

The tasks performed by DDL are:

- **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.
- **Truncate:** It is used to remove all records from a table.
- **Rename:** It is used to rename an object.
- **Comment:** It is used to comment on the data dictionary.
- These commands are used to update the database schema that's why they come under Data definition language.

Data Manipulation Language (DML)

- **DML** stands for **D**ata **M**anipulation **L**anguage.
- It is used for accessing and manipulating data in a database.
- It handles user requests.

The tasks performed by DDL are:

- **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 all records from a table.

Data Manipulation Language (DML)

- **Merge:** It performs UPSERT operation, i.e., insert or update operations.
- **Call:** It is used to call a structured query language or a Java subprogram.
- **Explain Plan:** It has the parameter of explaining data.
- **Lock Table:** It controls concurrency.

Data Control Language (DCL)

- **DCL** stands for **D**ata **C**ontrol **L**anguage. It is used to retrieve the stored or saved data.
- The DCL execution is transactional. It also has rollback parameters.
- (But in Oracle database, the execution of data control language does not have the feature of rolling back.)

The tasks performed by DCL are:

- **Grant:** used to give user access privileges to a database.
- **Revoke:** It is used to take back permissions from the user.
- There are the following operations which have the authorization of Revoke:

  CONNECT, INSERT, USAGE, EXECUTE, DELETE, UPDATE and SELECT.

Transaction Control Language (TCL)

- TCL is used to run the changes made by the DML statement.
- TCL can be grouped into a logical transaction.

The tasks performed by TCL are:

- **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.

Data Sub Language (DSL)

- It is a combination of both DML and DDL.

DSL = DDL + DML

Storage Definition Language (SDL)

- It is defined as a language used to specify the internal schema in the database.
- In SDL, the storage structure and access methods used by the database system is specified by a set of statements.
- These statements define the implementation details of the database schema which are usually hidden from the users.

View Definition Language (VDL)

- It is defined as a language used to specify user's views (external schema) and their mappings to the conceptual schema.

- DDL is used to specify both the conceptual and external schemas.

- There are two views of data:
  - ✓Logical View of Data: It is the form that the programmer perceives to be in.
  - ✓Physical view of data: It is the form in which the data is actually stored on storage devices like disk.

Host Language (HL)

- It is defined as a language in which DML commands are embedded.
- Many DBMS have a facility for embedding SQL queries in high-level programming languages such as VB 6.0, VC++ 6.0 etc.

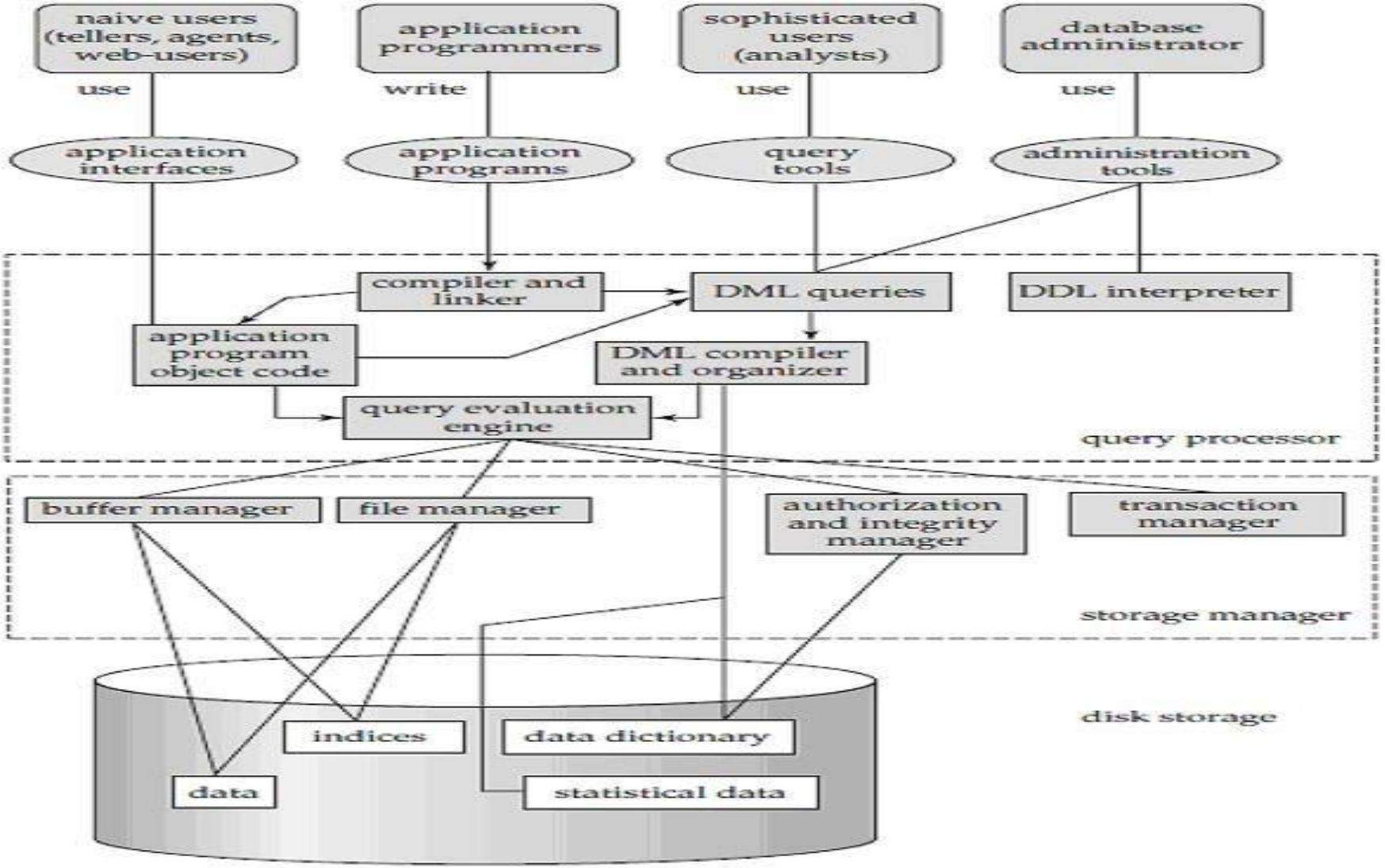
Fourth-Generation Languages (4GL)

- It is compact, efficient and non-procedural programming language that is used to improve the productivity of the DBMS.
- Examples of 4GLs are:
  - ✓ SQL
  - ✓ Report Generators
  - ✓ Spreadsheets
  - ✓ Code Generators
  - ✓ Automated Generation of HTML

Overall Database Structure

- A database system is partitioned into modules that deal with each of the responsibilities of the overall system.
- The functional components of a database system can be broadly divided into the **storage manager** and the **query processor** components.
- The storage manager is important because databases typically require a large amount of storage space.
- The query processor is important because it helps the database system simplify and facilitate access to data.

Overall Database Structure



Overall Database Structure

Query Processor include

- ✓ **DDL interpreter,** which interprets DDL statements and records the definitions in the data dictionary.
- ✓ **DML compiler,** which translates DML statements in a query language written into an evaluation plan consisting of low-level instructions that the query evaluation engine understands.

A query can usually be translated into any of several alternative evaluation plans that all give the same result. The DML compiler also performs **query optimization**, that is, it picks the lowest cost evaluation plan among the alternatives.

- ✓ **Query evaluation engine,** which executes low-level instructions generated by the DML compiler.

Overall Database Structure

Storage Manager

- A *storage manager* is a program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.
- The storage manager is responsible for the interaction with the file manager.
- The raw data are stored on the disk using the file system, which is usually provided by a conventional operating system.
- The storage manager translates the various DML statements into low-level file-system commands.
- Thus, the storage manager is responsible for storing, retrieving, and updating data in the database.

Overall Database Structure

- The storage manager components include:
  - ✓ **Authorization and integrity manager:** tests for the satisfaction of integrity constraints and checks the authority of users to access data.
  - ✓ **Transaction manager:** ensures that the database remains in a consistent state despite of system failures, and that concurrent transaction executions proceed without conflicting.
  - ✓ **File manager:** manages the allocation of space on disk and the data structures used to represent information stored on disk.
  - ✓ **Buffer manager:** is responsible for fetching data from disk storage into main memory and deciding what data to cache in main memory. The buffer manager is a critical part of the database system, since it enables the database to handle data sizes that are much larger than the size of main memory.

Overall Database Structure

- A **transaction** is a collection of operations that performs a single logical function in a database application.
- Each transaction is a unit of both atomicity and consistency.
- Thus, we require that transactions do not violate any database-consistency constraints.
- That is, if the database was consistent when a transaction started, the database must be consistent when the transaction successfully terminates.
- **Transaction - manager** ensures that the database remains in a consistent (correct) state despite of system failures (e.g., power failures and operating system crashes) and transaction failures.

Entity

- An entity can be a real-world object, that can be easily identifiable.
- For example, in a school database, students, teachers and courses offered can be considered as entities.
- All these entities have some attributes or properties that give them their identity.
- An entity set is a collection of similar types of entities. An entity set may contain entities with attribute sharing similar values.
- For example, a Students set may contain all the students of University, similarly a Teachers set may contain all the teachers of University from all faculties.

Attributes

- Entities are represented by means of their properties, called as **attributes**.
- All attributes have values. For example, a student entity may have name, class, and age as attributes.
- There exists a domain or range of values that can be assigned to attributes.
- For example, a student's name cannot be a numeric value. It has to be alphabetic.
- A student's age cannot be negative, etc.

Type of Attributes

- **Simple attribute:** Simple attributes are atomic values, which cannot be divided further. For example, a student's phone number is an atomic value of 10 digits.
- **Composite attribute:** Composite attributes are made of more than one simple attribute. For example, a student's complete name may have first name and last name.
- **Derived attribute:** Derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database. For example, average_salary in a department should not be saved directly in the database, instead it can be derived.
- **Single-value attribute:** Single-value attributes contain single value. For example: Social_Security_Number.
- **Multi-value attribute:** Multi-value attributes may contain more than one values. For example, a person can have more than one phone number, email_address, etc.

Type of Attributes

These attribute types can come together in a way like:

- ✓Simple single-valued attributes
- ✓Simple multi-valued attributes
- ✓Composite single-valued attributes
- ✓Composite multi-valued attributes

Keys

- Key is an attribute or collection of attributes that uniquely identifies an entity among entity set.
- For example, the roll_number of a student makes him/her identifiable among students.
- ✓ **Super Key** − A set of attributes (one or more) that collectively identifies an entity in an entity set.
- ✓ **Candidate Key** − A minimal super key is called a candidate key. An entity set may have more than one candidate key.
- ✓ **Primary Key** − A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.

Relationship

- The association among entities is called a relationship. For example, an employee **works_at** a department, a student **enrolls** in a course. Here, Works_at and enrolls are called relationships.

Relationship Set

- A set of relationships of similar type is called a relationship set. Like entities, a relationship too can have attributes. These attributes are called **descriptive attributes**.

Degree of Relationship

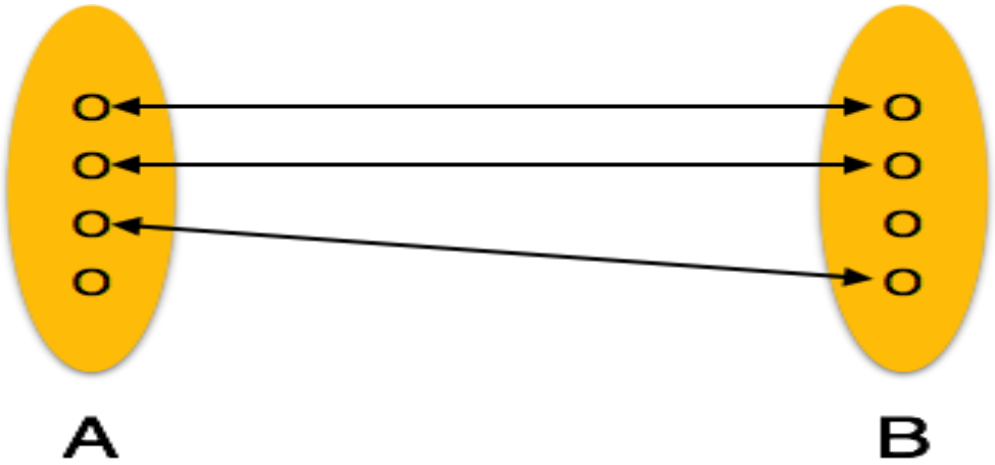
- The number of participating entities in a relationship defines the degree of the relationship.
- For example, Binary = degree 2,Ternary = degree 3

Relationship

Mapping Cardinalities

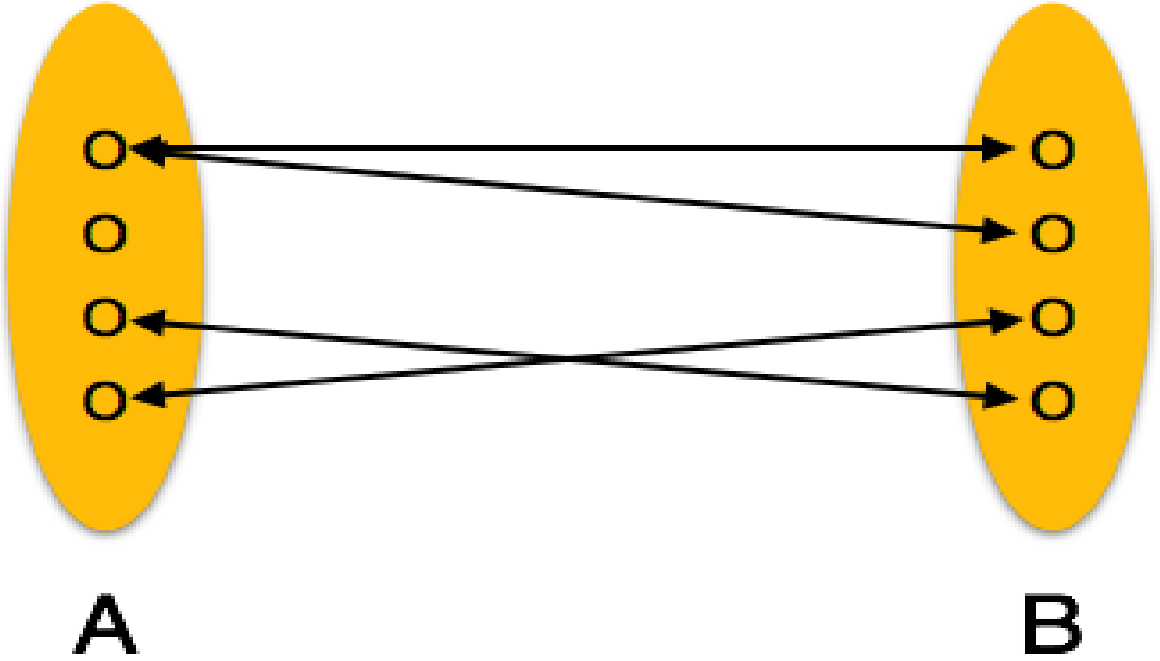
➢**Cardinality** defines the number of entities in one entity set, which can be associated with the number of entities of other set via relationship set.

✓**One-to-one** − One entity from entity set A can be associated with at most one entity of entity set B and vice versa.



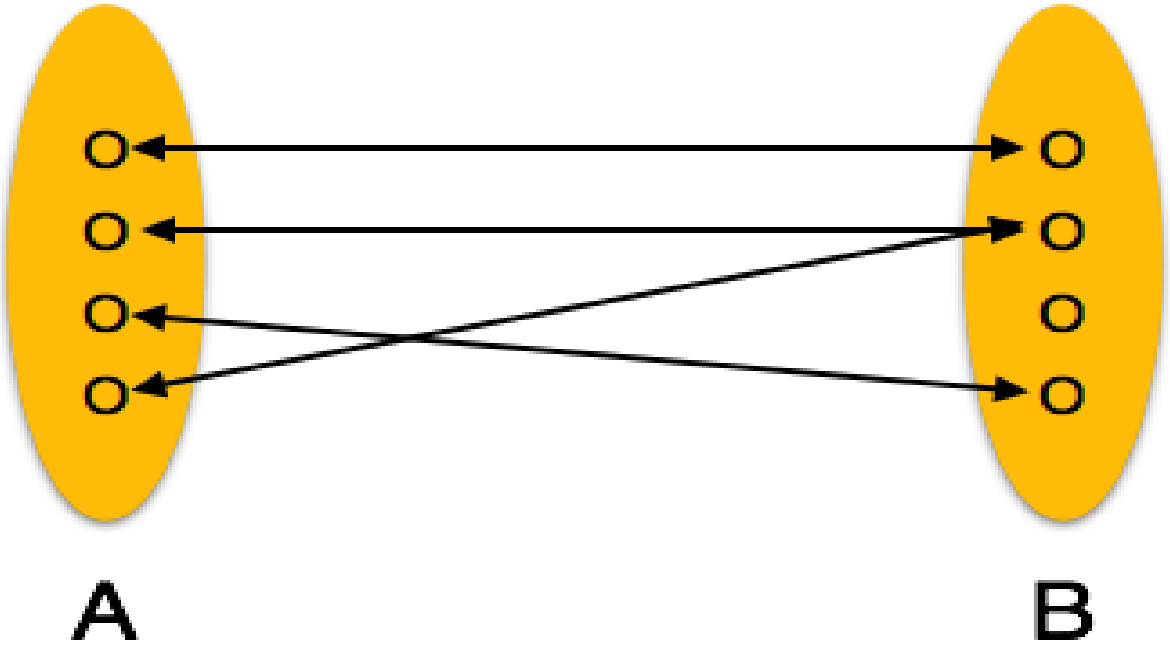
Relationship

- ✓ **One-to-many** − One entity from entity set A can be associated with more than one entities of entity set B however an entity from entity set B, can be associated with at most one entity.



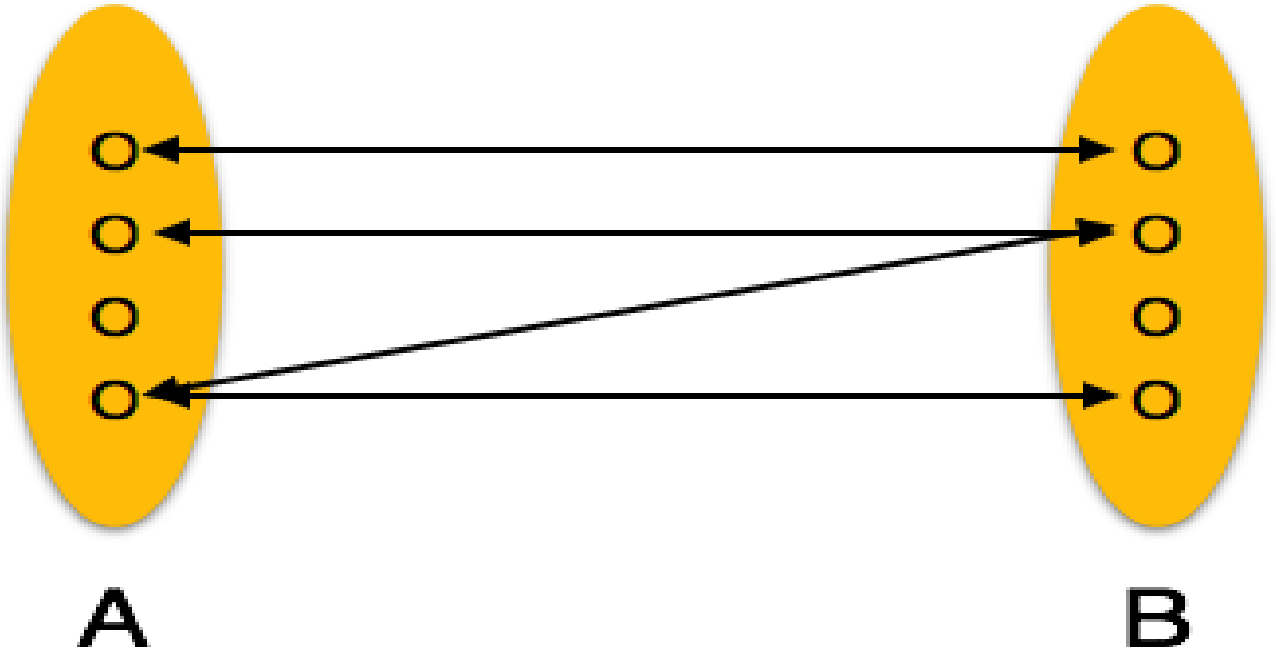
Relationship

- ✓ **Many-to-one** − More than one entities from entity set A can be associated with at most one entity of entity set B, however an entity from entity set B can be associated with more than one entity from entity set A.



Relationship

- ✓ **Many-to-many** − One entity from A can be associated with more than one entity from B and vice versa.



Entity- Relationship Diagram

ER Model is represented by means of an ER diagram. Any object, for example, entities, attributes of an entity, relationship sets, and attributes of relationship sets, can be represented with the help of an ER diagram.

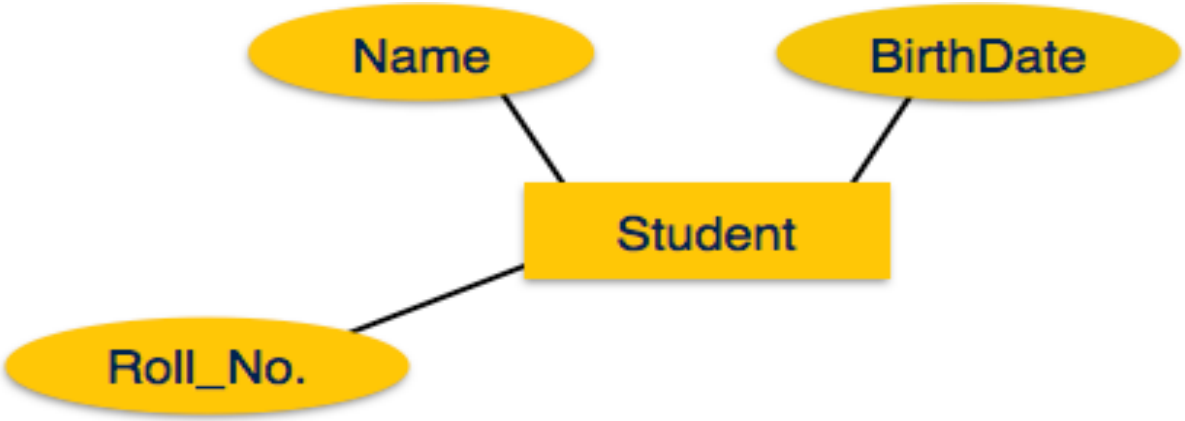
Entity

Entities are represented by means of rectangles. Rectangles are named with the entity set they represent.

Entity- Relationship Diagram

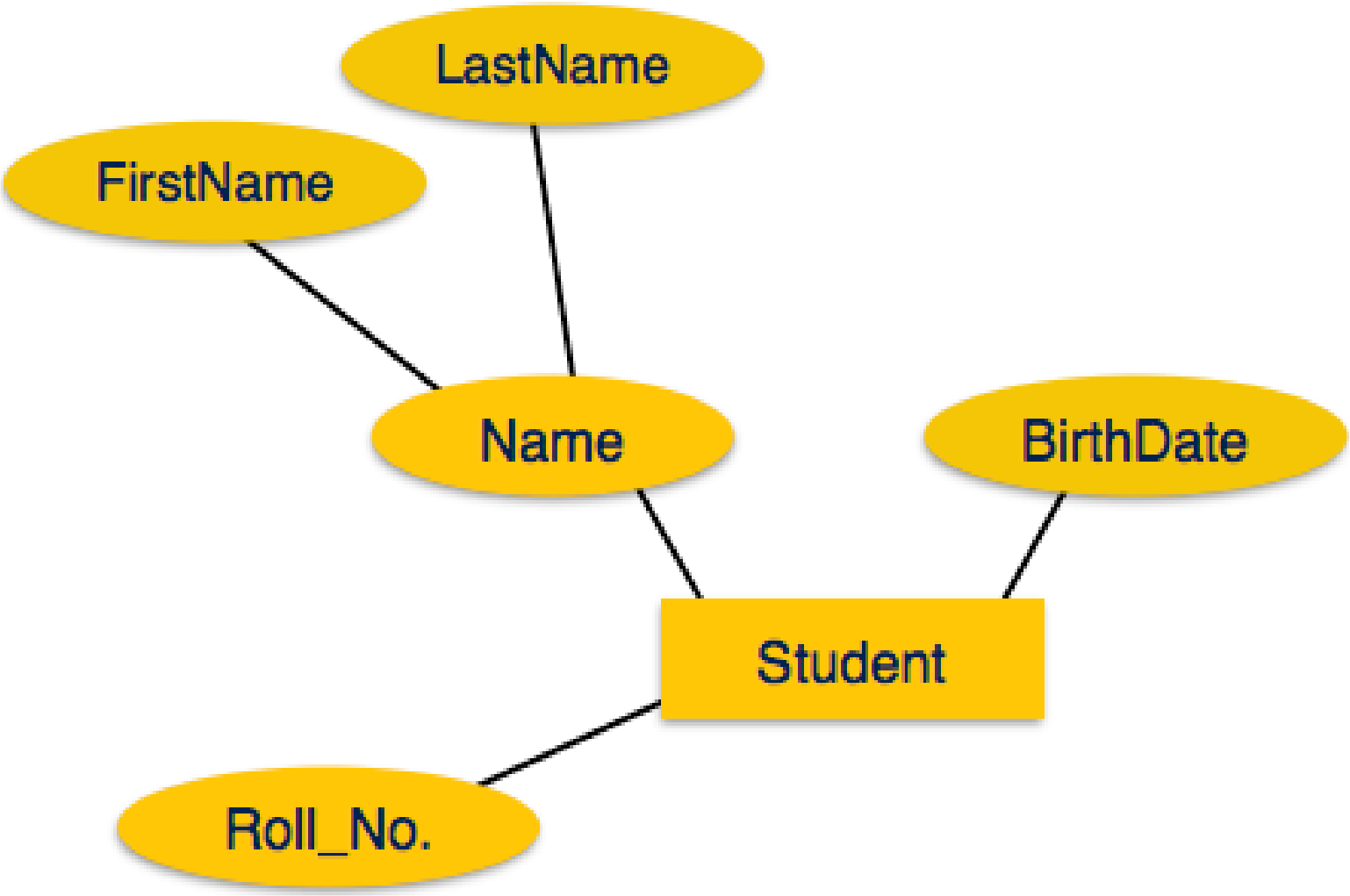
Attributes

Attributes are the properties of entities. Attributes are represented by means of ellipses. Every ellipse represents one attribute and is directly connected to its entity (rectangle).



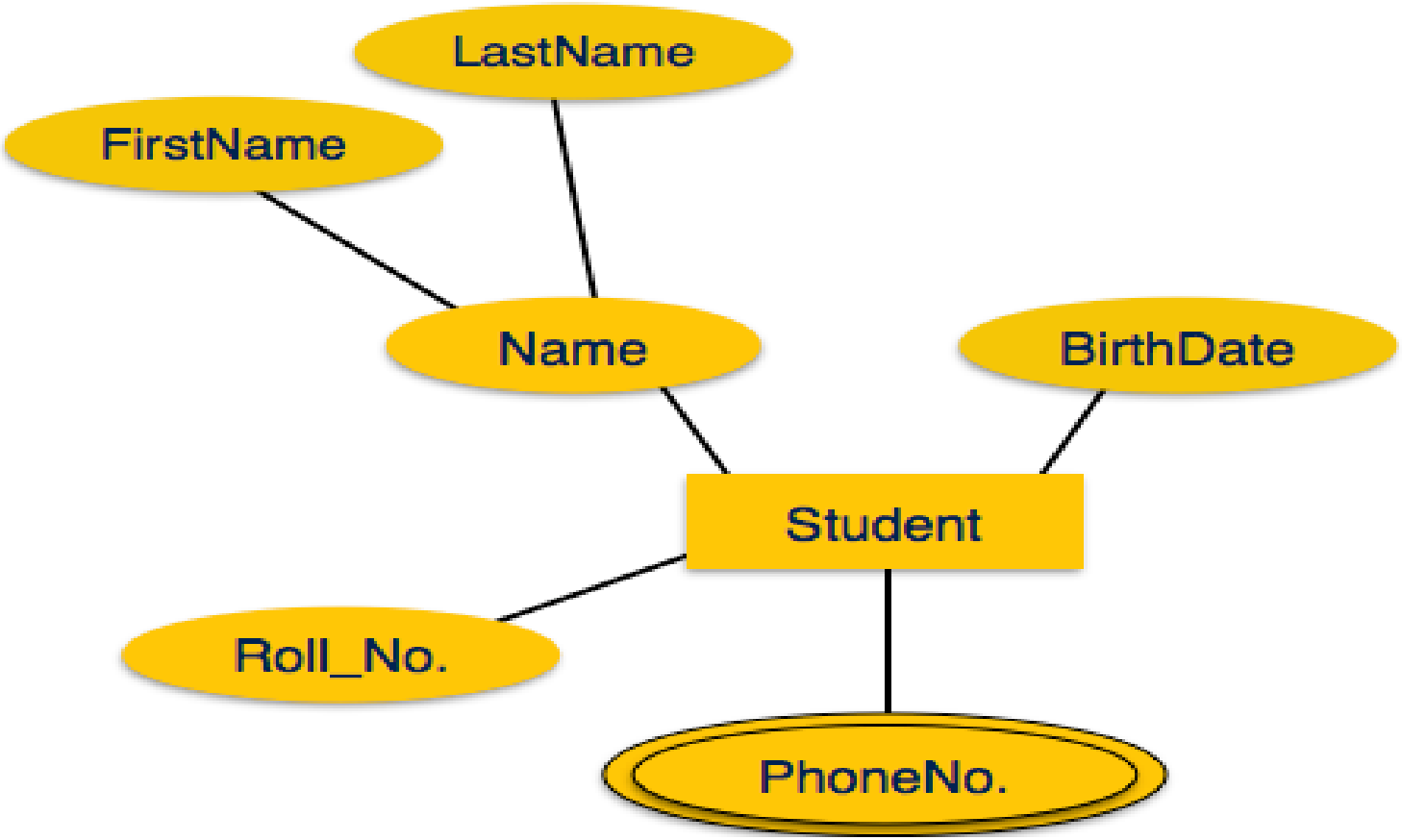
If the attributes are composite, they are further divided in a tree like structure. Every node is then connected to its attribute. That is, composite attributes are represented by ellipses that are connected with an ellipse.

Entity- Relationship Diagram



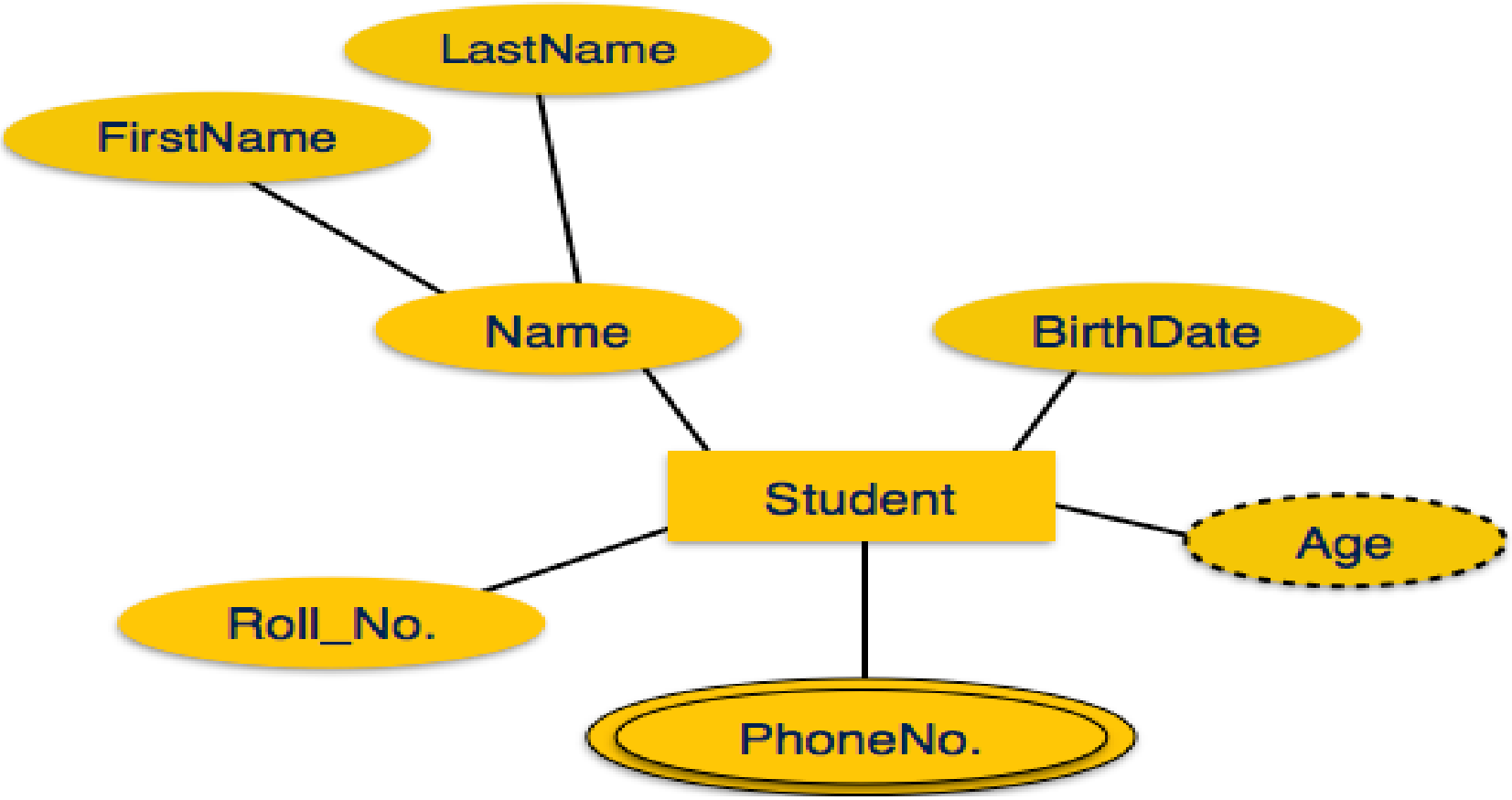
Entity- Relationship Diagram

Multivalued attributes are depicted by double ellipse.



Entity- Relationship Diagram

Derived attributes are depicted by dashed ellipse.



Entity- Relationship Diagram

Relationship

❖Relationships are represented by diamond-shaped box.

❖ Name of the relationship is written inside the diamond-box.

❖ All the entities (rectangles) participating in a relationship, are connected to it by a line.

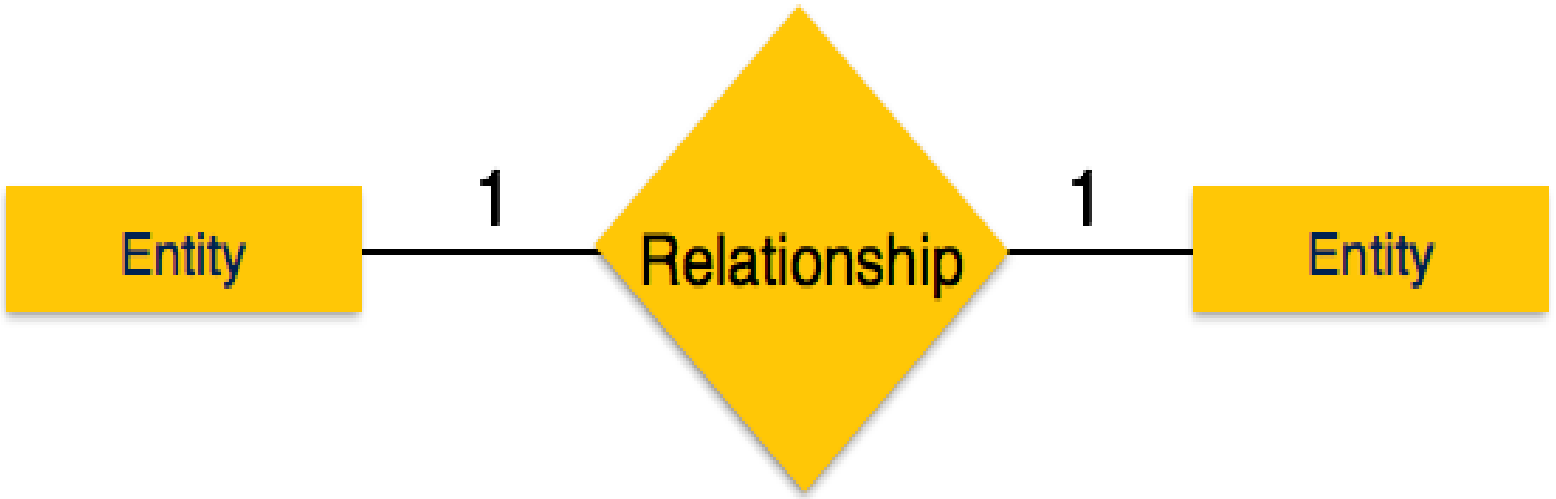
Binary Relationship and Cardinality

❖A relationship where two entities are participating is called a binary relationship.

❖Cardinality is the number of instance of an entity from a relation that can be associated with the relation.

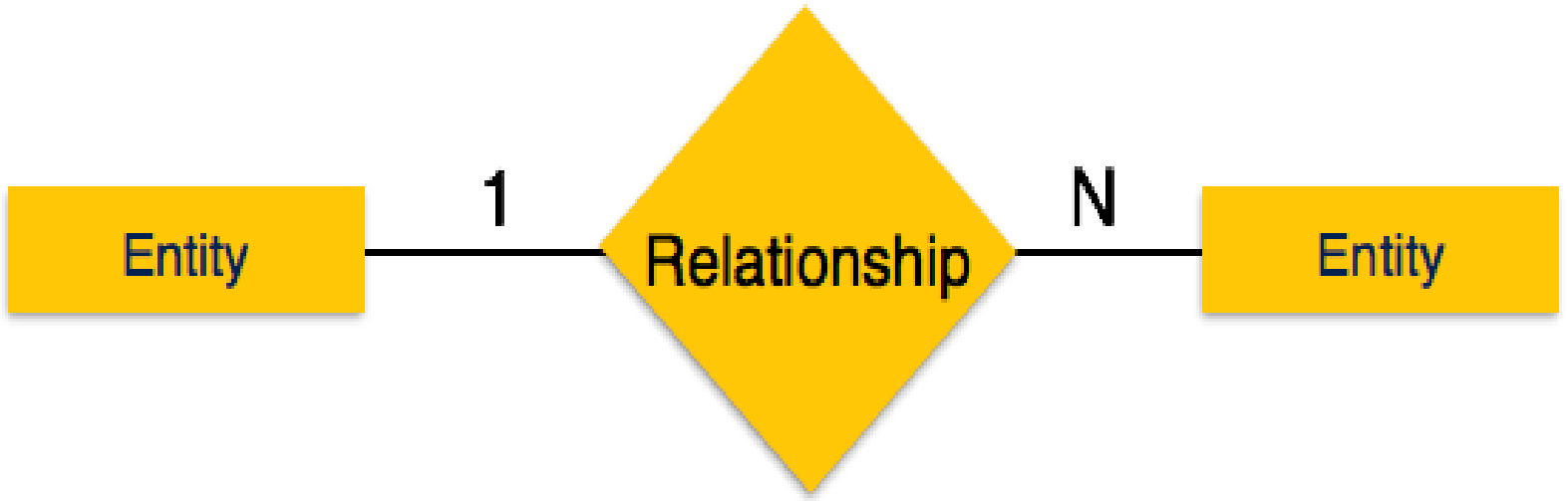
Entity- Relationship Diagram

- ✓ One-to-one − When only one instance of an entity is associated with the relationship, it is marked as '1:1'. The following image reflects that only one instance of each entity should be associated with the relationship. It depicts one-to-one relationship.



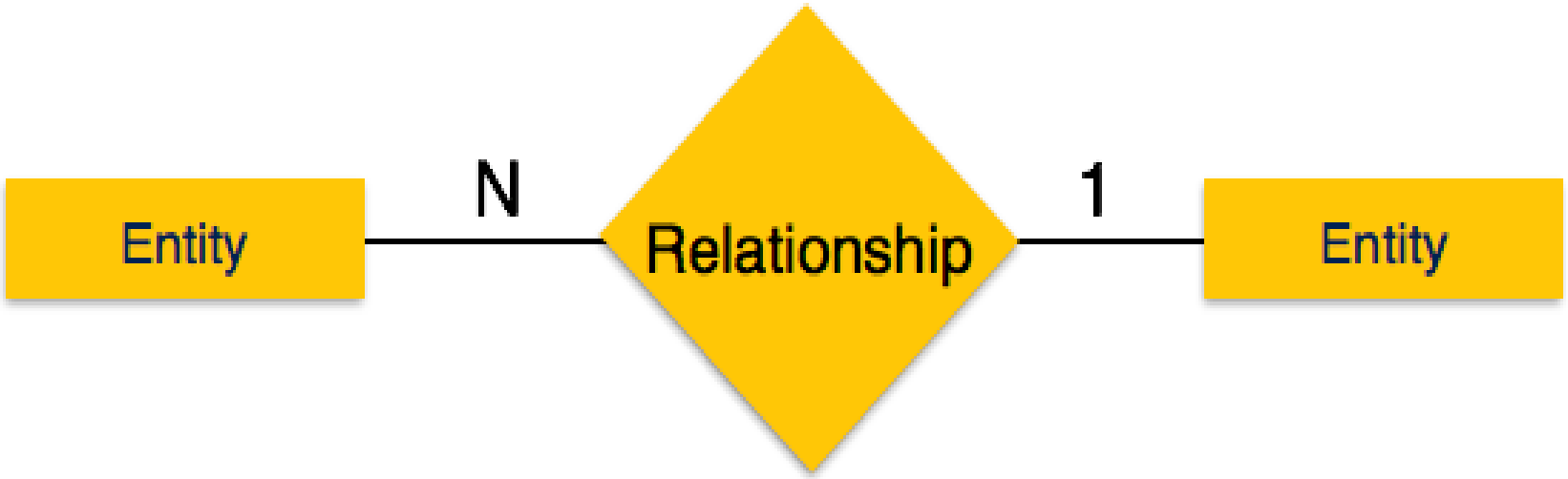
Entity- Relationship Diagram

- ✓ One-to-many − When more than one instance of an entity is associated with a relationship, it is marked as '1:N'. The following image reflects that only one instance of entity on the left and more than one instance of an entity on the right can be associated with the relationship. It depicts one-to-many relationship.



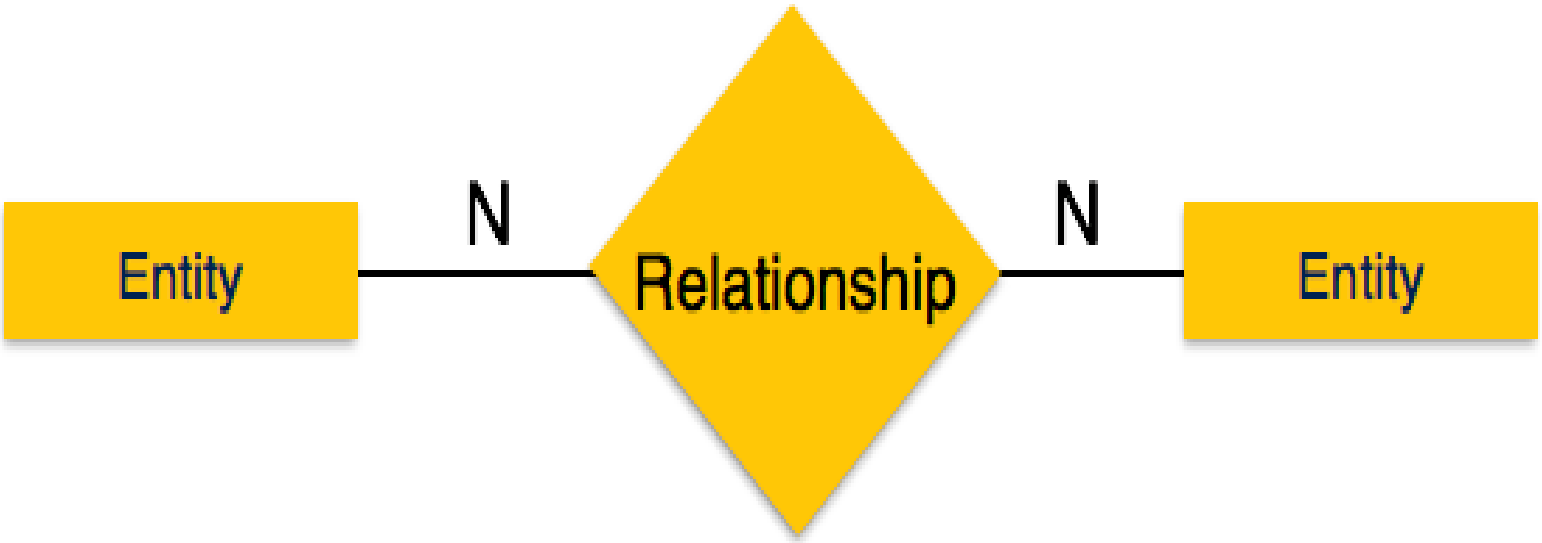
Entity- Relationship Diagram

- ✓ Many-to-one − When more than one instance of entity is associated with the relationship, it is marked as 'N:1'. The following image reflects that more than one instance of an entity on the left and only one instance of an entity on the right can be associated with the relationship. It depicts many-to-one relationship.



Entity- Relationship Diagram

- ✓ Many-to-many − The following image reflects that more than one instance of an entity on the left and more than one instance of an entity on the right can be associated with the relationship. It depicts many-to-many relationship.

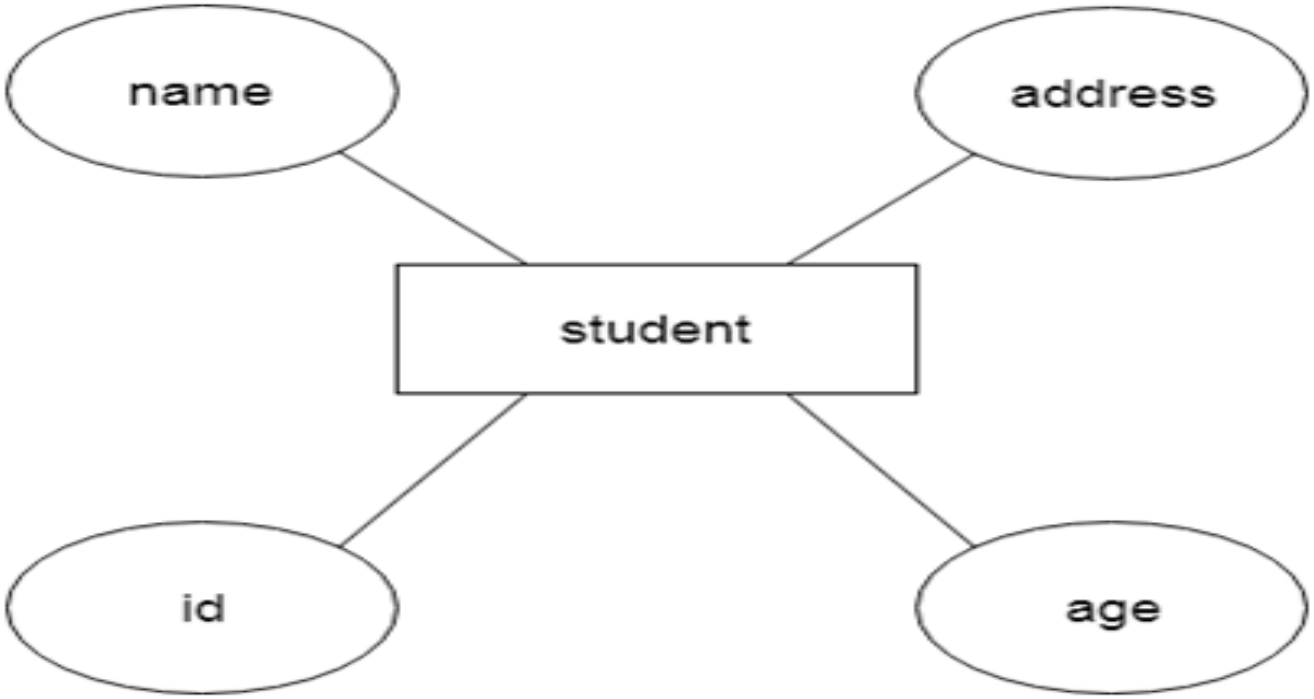


E-R model

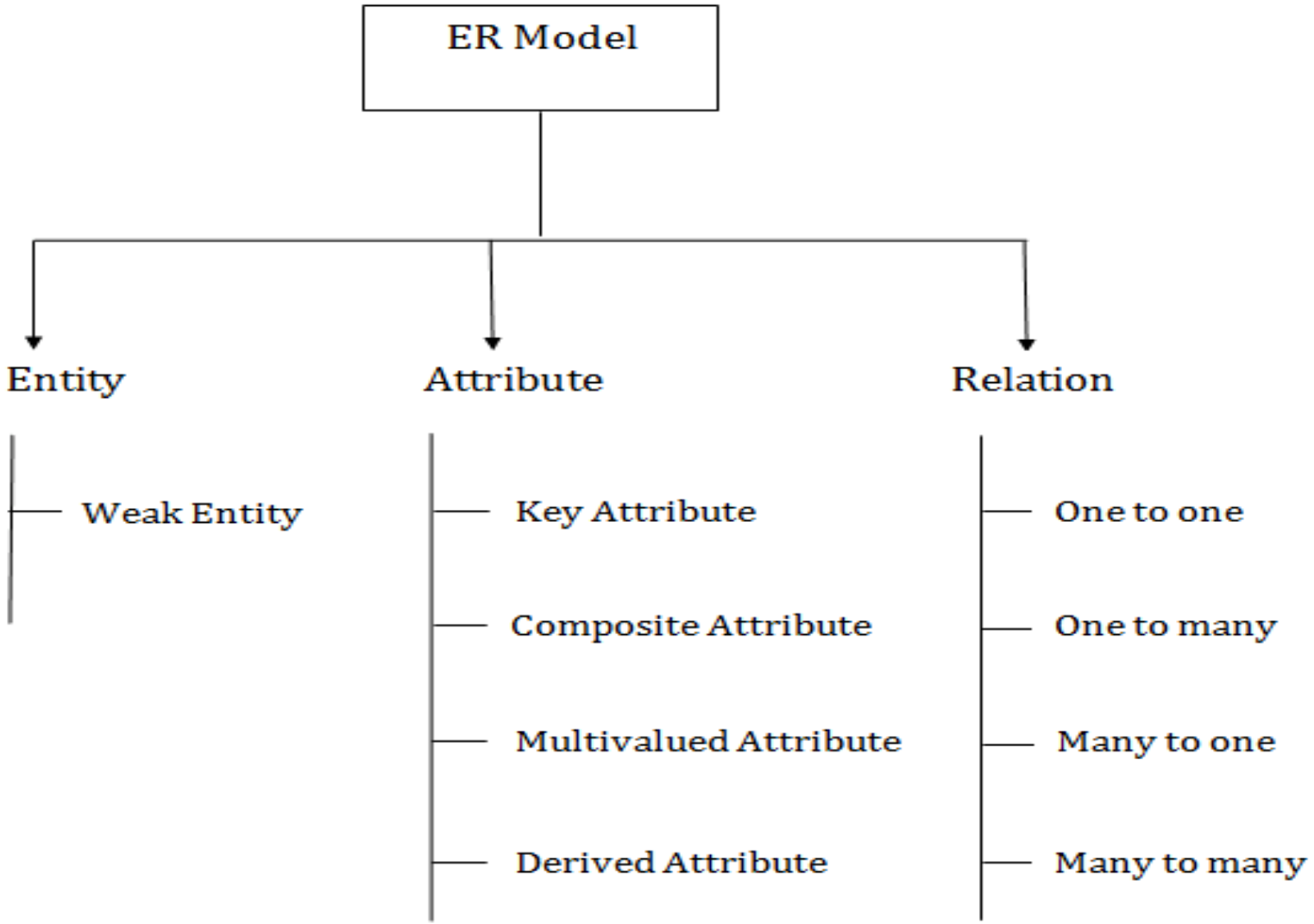
- ER model stands for an Entity-Relationship model.
- It is a high-level data model.
- This model is used to define the data elements and relationship for a specified system.
- It develops a conceptual design for the database.
- It also develops a very simple and easy to design view of data.
- In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

E-R model

For example, Suppose we design a school database. In this database, the student will be an entity with attributes like address, name, id, age, etc. The address can be another entity with attributes like city, street name, pin code, etc and there will be a relationship between them.

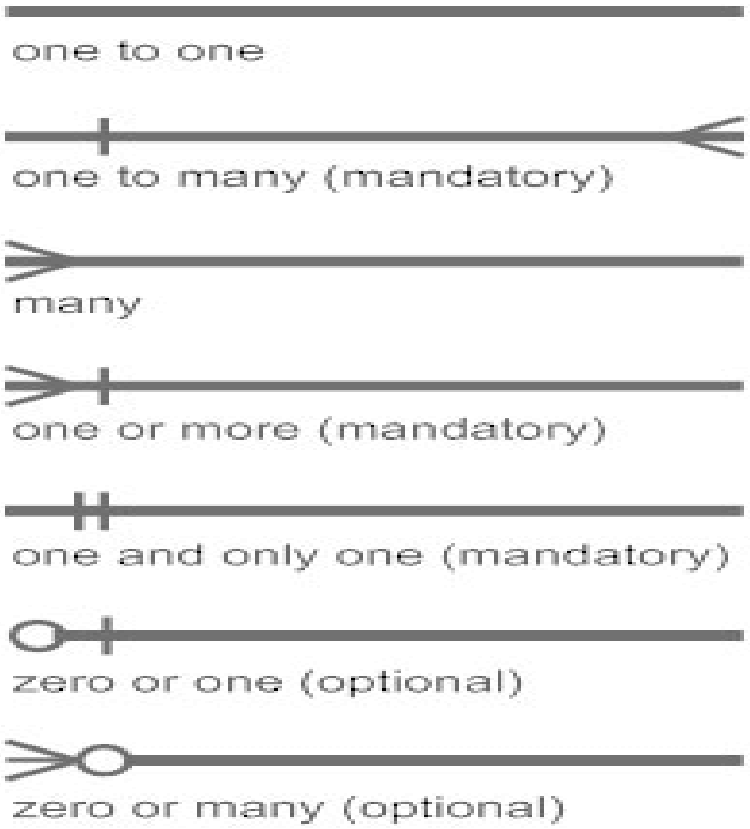


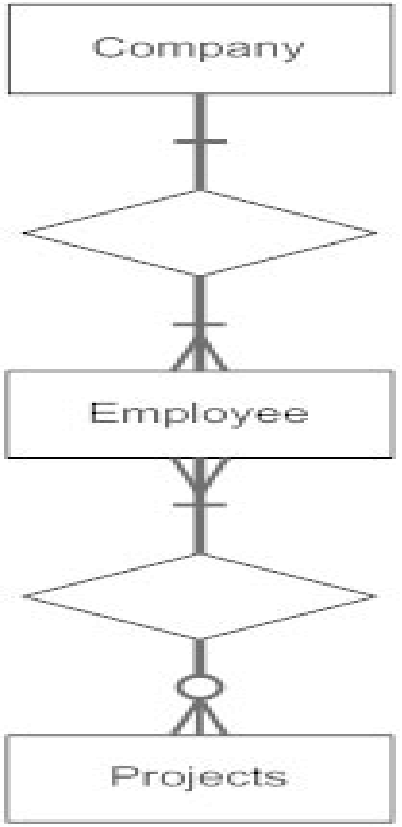
Components of E-R Diagram



Notation of ER diagram

Database can be represented using the notations. In ER diagram, many notations are used to express the cardinality. These notations are as follows:





Mapping Constraints

- A mapping constraint is a data constraint that expresses the number of entities to which another entity can be related via a relationship set.

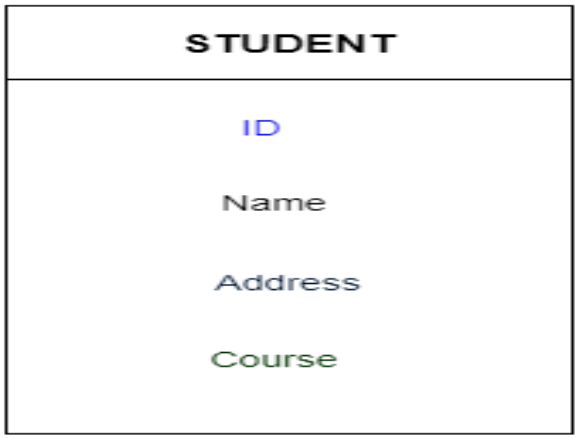
- It is most useful in describing the relationship sets that involve more than two entity sets.

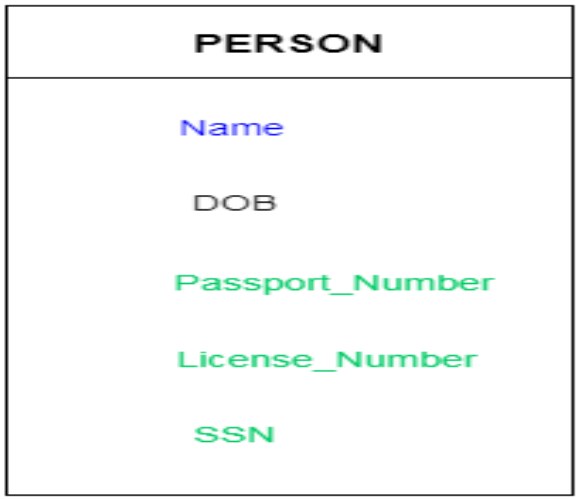
- For binary relationship set R on an entity set A and B, there are four possible mapping cardinalities. These are as follows:

  - ✓ One to one (1:1)
  - ✓ One to many (1:M)
  - ✓ Many to one (M:1)
  - ✓ Many to many (M:N)

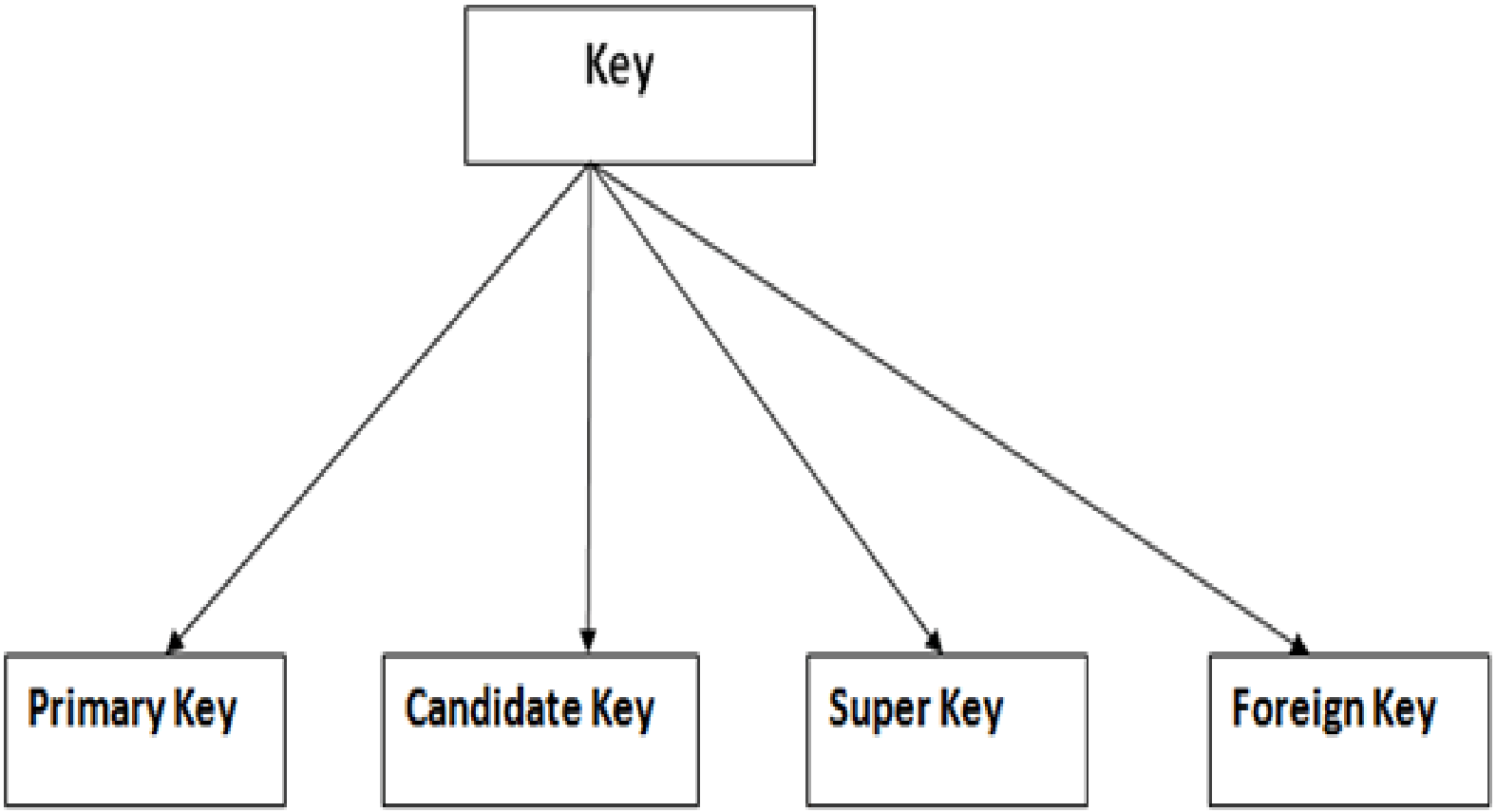
DBMS Keys

- 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.
- 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.





Types of key



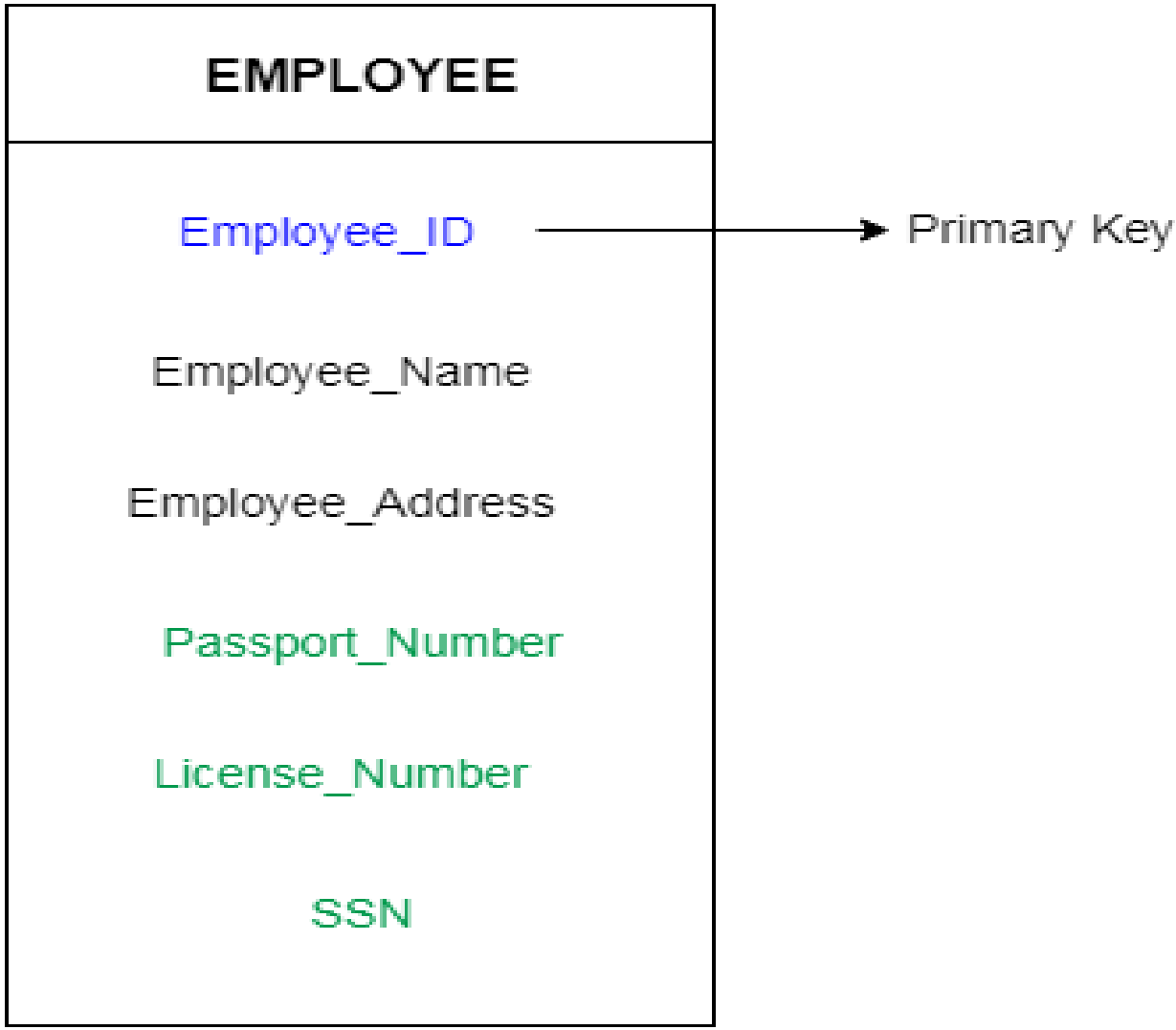
Primary key

- 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.

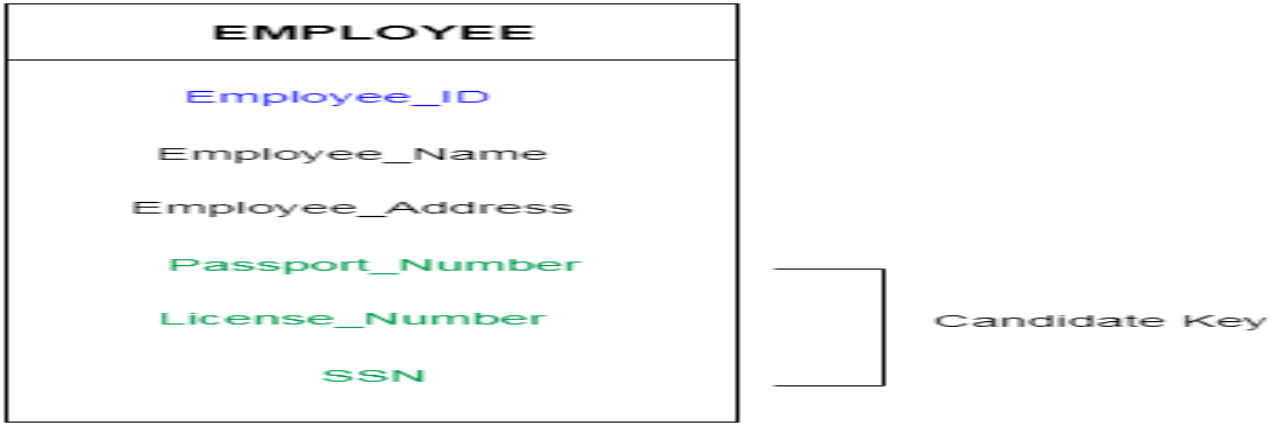
- For each entity, selection of the primary key is based on requirement of developers.

Primary key


EMPLOYEE
Employee_ID
Primary Key
Employee_Name
Employee_Address
Passport_Number
License_Number
SSN

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.



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.

- 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.

Foreign Key

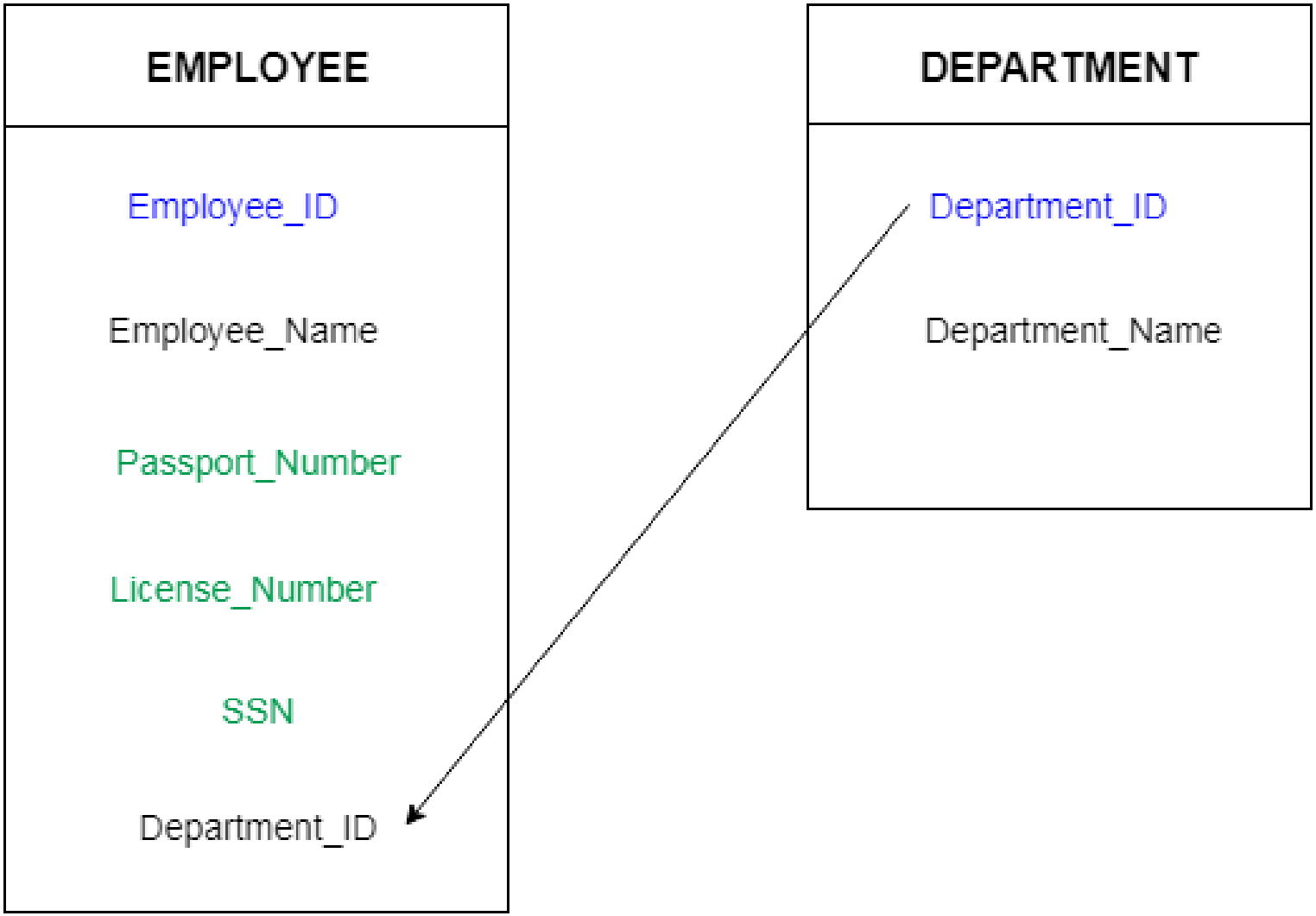
- Foreign keys are the column of the table which is used to point the primary key of another table.

- 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.

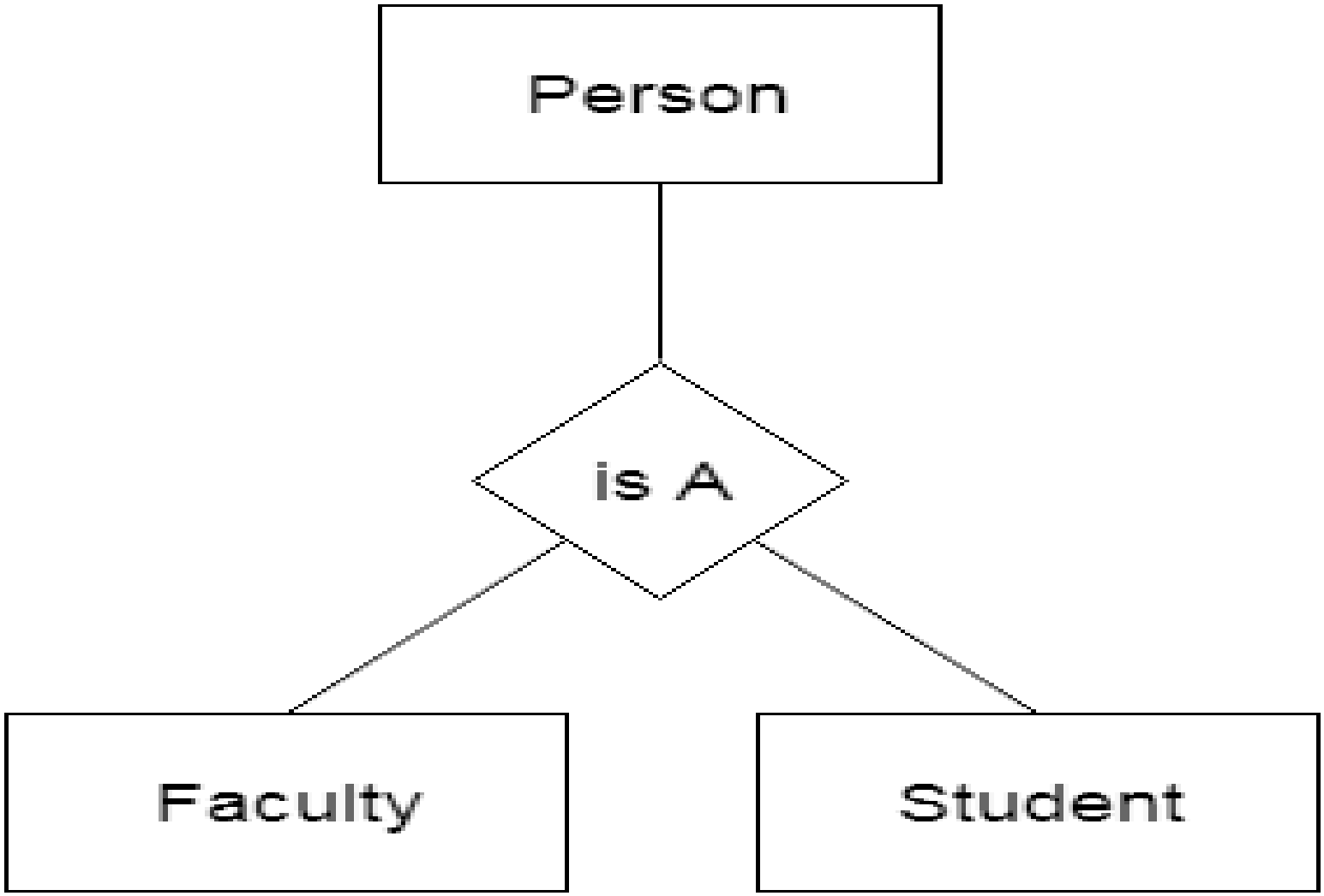
Foreign Key


EMPLOYEE
Employee_ID
Employee_Name
Passport_Number
License_Number
SSN
Department_ID
DEPARTMENT
Department_ID
Department_Name

Generalization

- Generalization is like a bottom-up approach in which two or more entities of lower level combine to form a higher level entity if they have some attributes in common.
- In generalization, an entity of a higher level can also combine with the entities of the lower level to form a further higher level entity.
- Generalization is more like subclass and superclass system, but the only difference is the approach. Generalization uses the bottom-up approach.
- In generalization, entities are combined to form a more generalized entity, i.e., subclasses are combined to make a superclass.
- **For example,** Faculty and Student entities can be generalized and create a higher level entity Person.

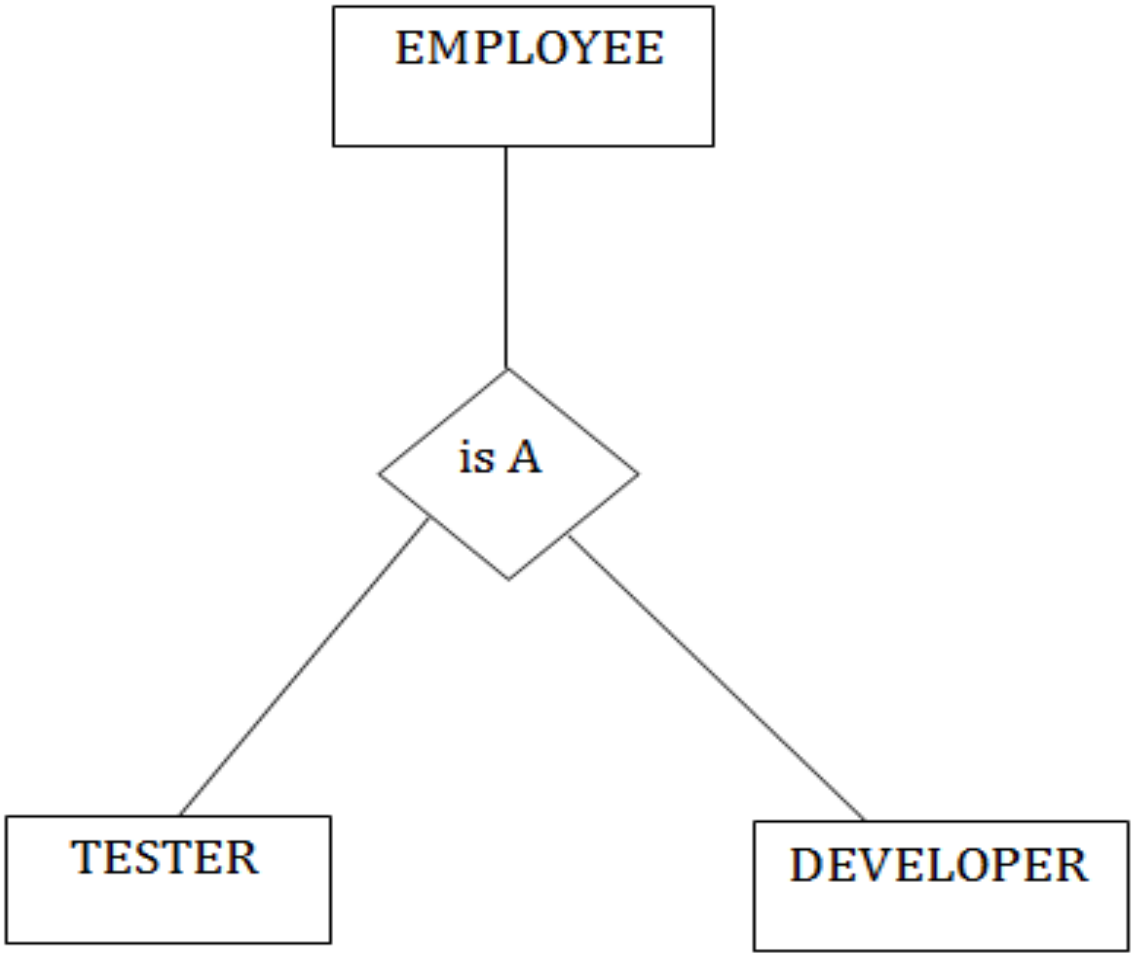
Generalization



Specialization

- Specialization is a top-down approach, and it is opposite to Generalization. In specialization, one higher level entity can be broken down into two lower level entities.
- Specialization is used to identify the subset of an entity set that shares some distinguishing characteristics.
- Normally, the superclass is defined first, the subclass and its related attributes are defined next, and relationship set are then added.
- In an Employee management system, EMPLOYEE entity can be specialized as TESTER or DEVELOPER based on what role they play in the company.

Specialization

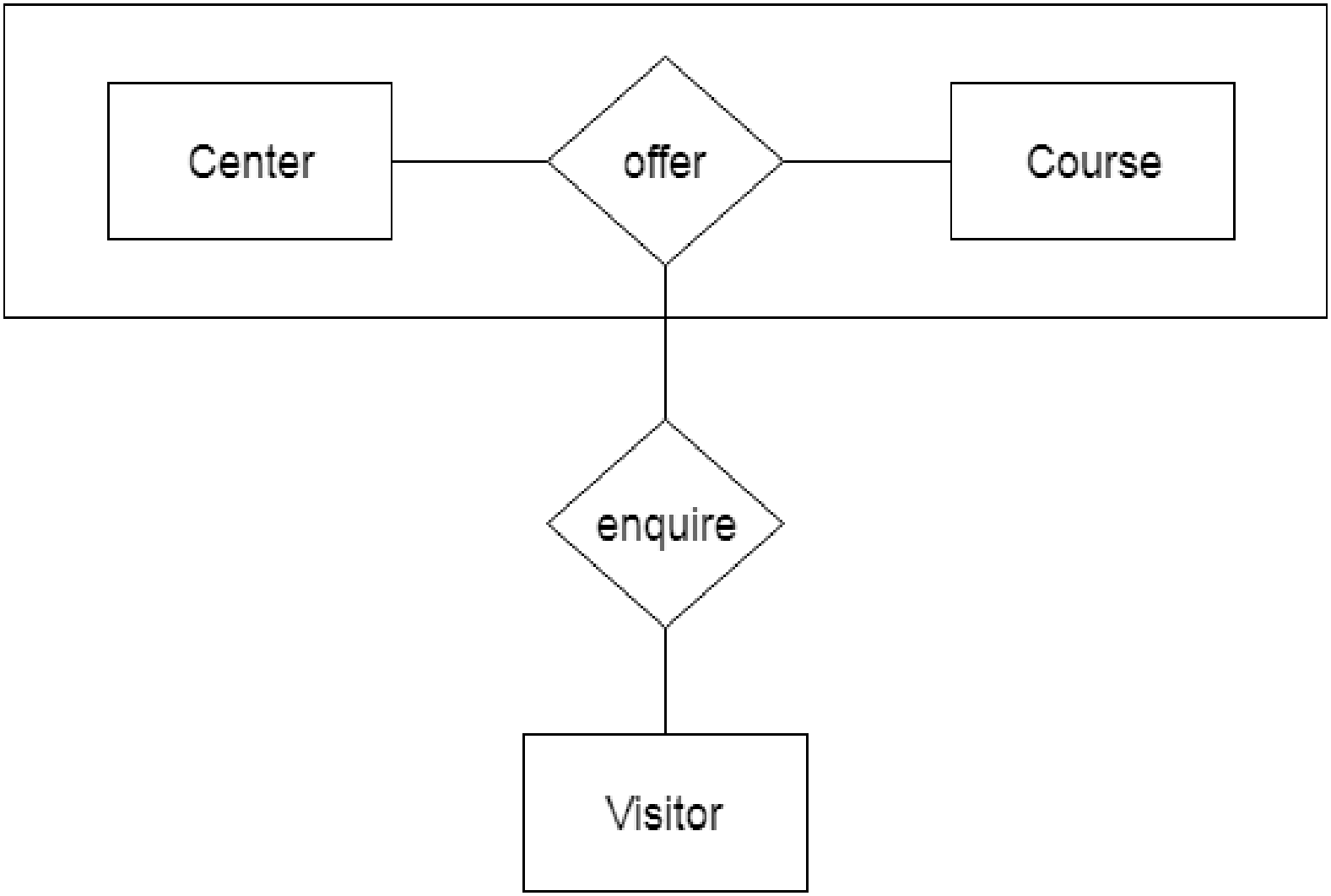


Aggregation

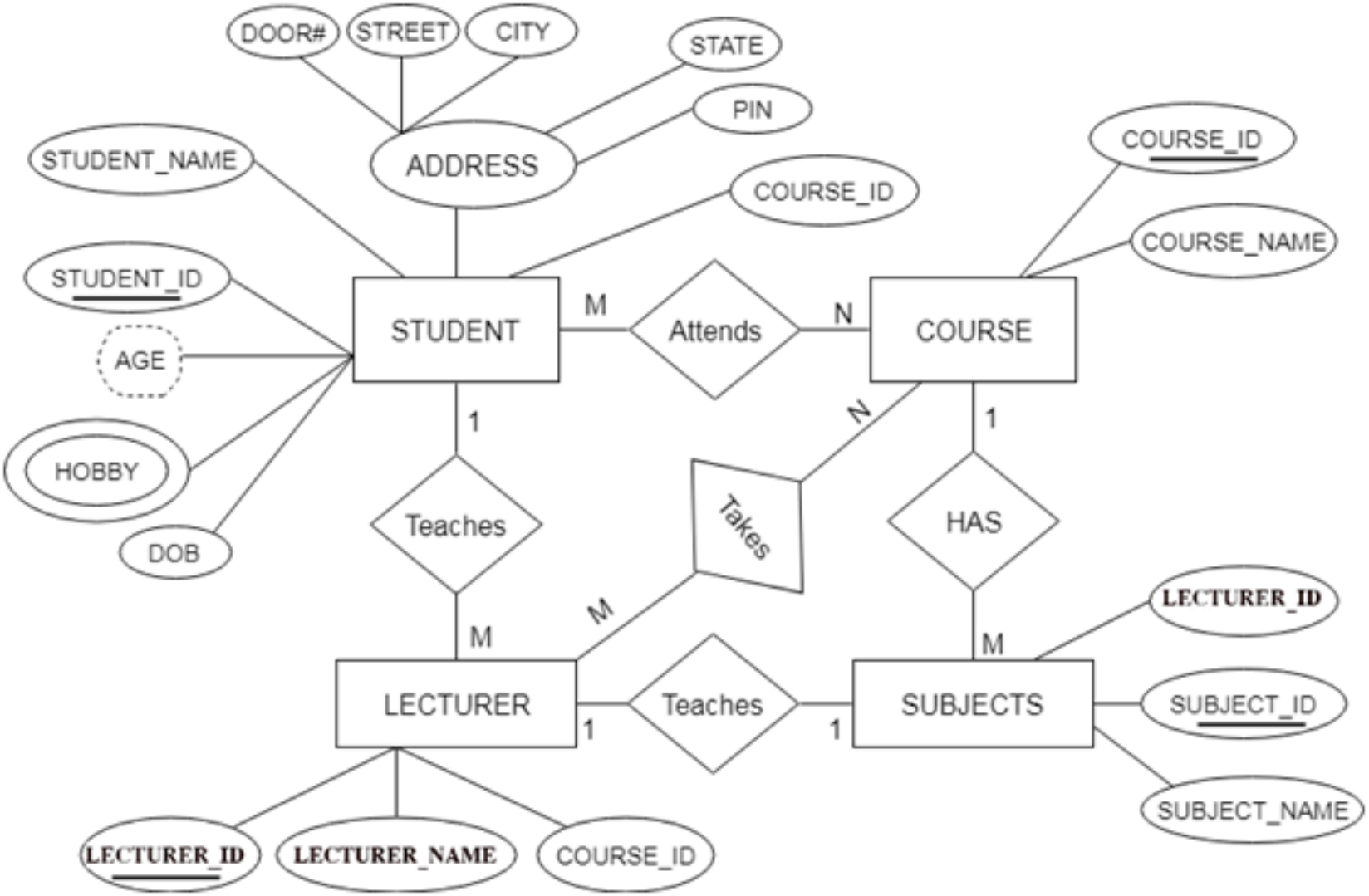
- In aggregation, the relation between two entities is treated as a single entity. In aggregation, relationship with its corresponding entities is aggregated into a higher level entity.

- Center entity offers the Course entity act as a single entity in the relationship which is in a relationship with another entity visitor. In the real world, if a visitor visits a coaching center then he will never enquiry about the Course only or just about the Center instead he will ask the enquiry about both.

Aggregation



Reduction of ER diagram to Table



Reduction of ER diagram to Table

- **Entity type becomes a table.**
- In the given ER diagram, LECTURER, STUDENT, SUBJECT and COURSE forms individual tables.
- **All single-valued attribute becomes a column for the table.**
- In the STUDENT entity, STUDENT_NAME and STUDENT_ID form the column of STUDENT table. Similarly, COURSE_NAME and COURSE_ID form the column of COURSE table and so on.
- **Key attribute of the entity type represented by the primary key.**

Reduction of ER diagram to Table

- In the given ER diagram, COURSE_ID, STUDENT_ID, SUBJECT_ID, and LECTURER_ID are the key attribute of the entity.

- **The multivalued attribute is represented by a separate table.**

- In the student table, hobby is a multivalued attribute. So it is not possible to represent multiple values in a single column of STUDENT table. Hence we create a table STUD_HOBBY with column name STUDENT_ID and HOBBY. Using both the column, we create a composite key.

- **Composite attribute represented by components.**

Reduction of ER diagram to Table

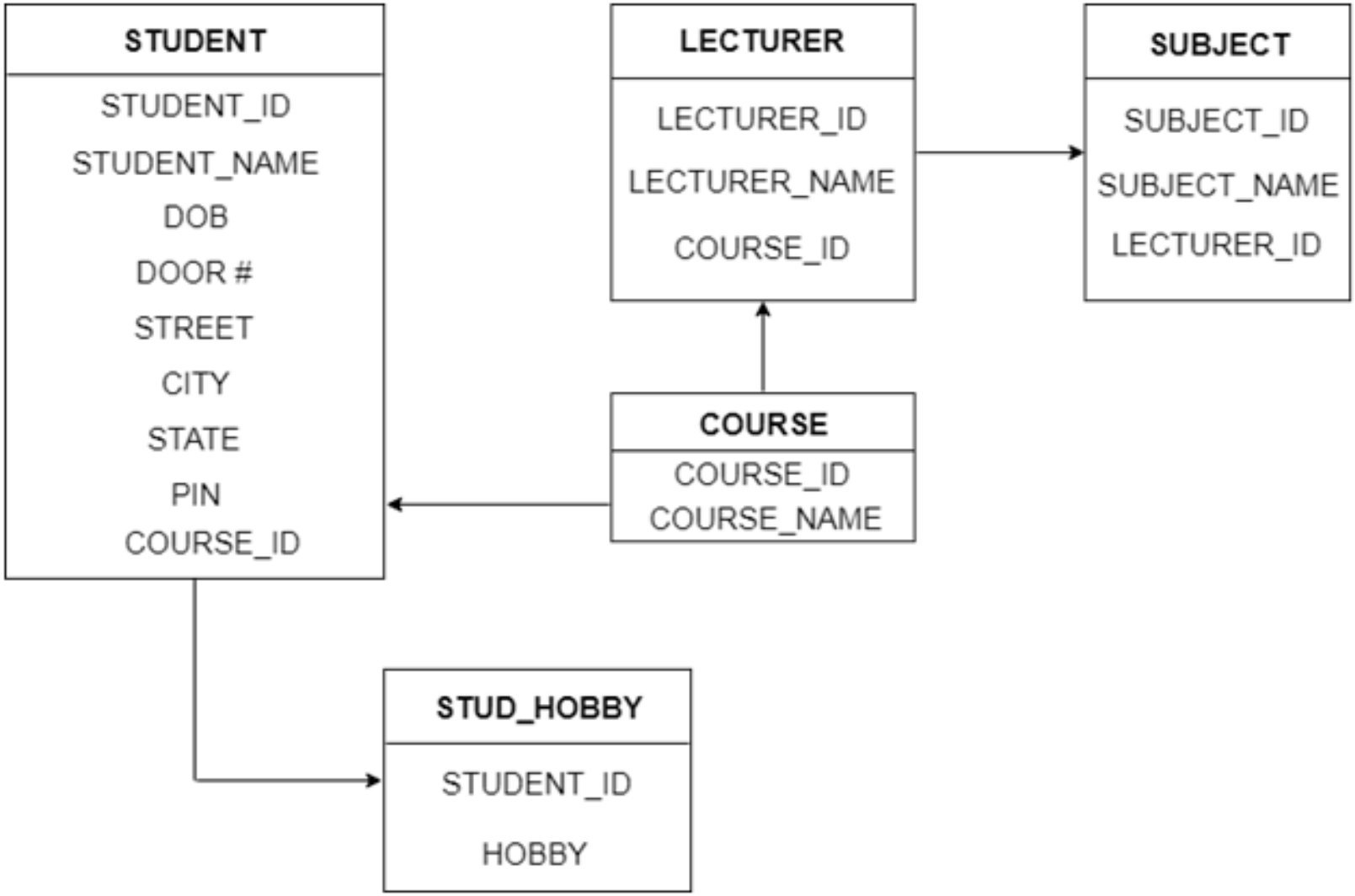
- In the given ER diagram, student address is a composite attribute. It contains CITY, PIN, DOOR#, STREET, and STATE. In the STUDENT table, these attributes can merge as an individual column.

- **Derived attributes are not considered in the table.**

- In the STUDENT table, Age is the derived attribute. It can be calculated at any point of time by calculating the difference between current date and Date of Birth.

- Using these rules, We can convert the ER diagram to tables and columns and assign the mapping between the tables. Table structure for the given ER diagram is as below:

Reduction of ER diagram to Table



Relationship of Higher Degree

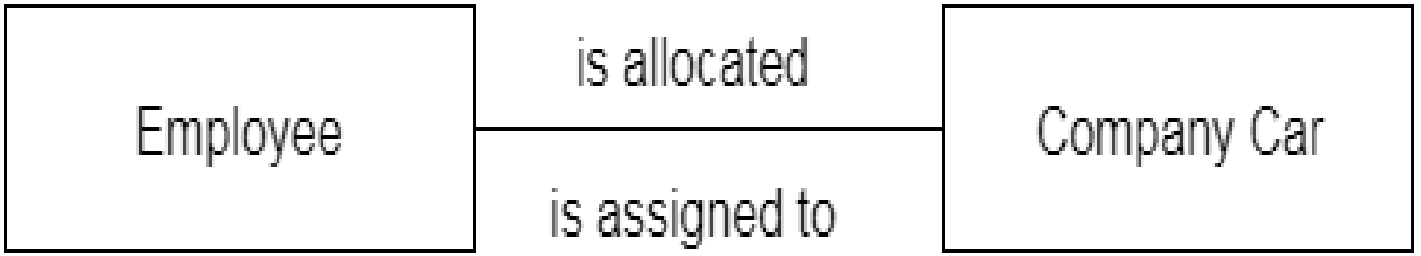
- The degree of relationship can be defined as the number of occurrences in one entity that is associated with the number of occurrences in another entity.

- There are the three degree of relationship:
  - ✓ One-to-one (1:1)
  - ✓ One-to-many (1:M)
  - ✓ Many-to-many (M:N)

Relationship of Higher Degree

One-to-one

- In a one-to-one relationship, one occurrence of an entity relates to only one occurrence in another entity.
- A one-to-one relationship rarely exists in practice.
- **For example:** if an employee is allocated a company car then that car can only be driven by that employee.
- Therefore, employee and company car have a one-to-one relationship.



Relationship of Higher Degree

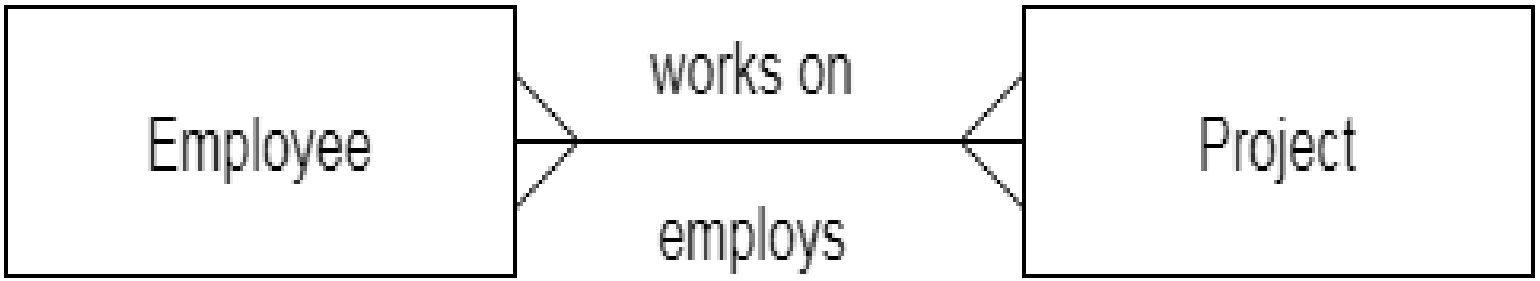
One-to-many

- In a one-to-many relationship, one occurrence in an entity relates to many occurrences in another entity.
- An employee works in one department, but a department has many employees.
- Therefore, department and employee have a one-to-many relationship.

Relationship of Higher Degree

Many-to-many

- In a many-to-many relationship, many occurrences in an entity relate to many occurrences in another entity.
- Same as a one-to-one relationship, the many-to-many relationship rarely exists in practice.
- At the same time, an employee can work on several projects, and a project has a team of many employees.
- Therefore, employee and project have a many-to-many relationship.



References

1. Date C J, “An Introduction To Database System”, Addision Wesley.
2. Korth, Silberchatz, Sudarshan, “Database Concepts”, McGraw Hill.
3. https://www.javatpoint.com/dbms-tutorial
4. https://www.tutorialspoint.com/dbms/index.htm