Database Management Systems

UNIT 1ST

PRASHANT TOMAR

- A **database** is an organized collection of data, generally stored and accessed electronically from a computer system. Where databases are more complex they are often developed using formal design and modeling techniques.
- **Database** is a collection of related data and data is a collection of facts and figures that can be processed to produce information.
- Mostly data represents recordable facts. Data aids in producing information, which is based on facts. For example, if we have data about marks obtained by all students, we can then conclude about toppers and average marks.

- The **database management system (DBMS)** is the software that interacts with end users, applications, and the database itself to capture and analyze the data.
- The DBMS software additionally encompasses the core facilities provided to administer the database. The sum total of the database, the DBMS and the associated applications can be referred to as a "database system". Often the term "database" is also used to loosely refer to any of the DBMS, the database system or an application associated with the database.
- A **database management system** stores data in such a way that it becomes easier to retrieve, manipulate, and produce information.

Characteristics

- Traditionally, data was organized in file formats. DBMS was a new concept then, and all the research was done to make it overcome the deficiencies in traditional style of data management. A modern DBMS has the following characteristics −
- **Real-world entity** − A modern DBMS is more realistic and uses real-world entities to design its architecture. It uses the behavior and attributes too. For example, a school database may use students as an entity and their age as an attribute.
- **Relation-based tables** − DBMS allows entities and relations among them to form tables. A user can understand the architecture of a database just by looking at the table names.

- **Isolation of data and application** − A database system is entirely different than its data. A database is an active entity, whereas data is said to be passive, on which the database works and organizes. DBMS also stores metadata, which is data about data, to ease its own process.
- **Less redundancy** − DBMS follows the rules of normalization, which splits a relation when any of its attributes is having redundancy in values. Normalization is a mathematically rich and scientific process that reduces data redundancy.
- **Consistency** − Consistency is a state where every relation in a database remains consistent. There exist methods and techniques, which can detect attempt of leaving database in inconsistent state. A DBMS can provide greater consistency as compared to earlier forms of data storing applications like file-processing systems.

- **Query Language** − DBMS is equipped with query language, which makes it more efficient to retrieve and manipulate data. A user can apply as many and as different filtering options as required to retrieve a set of data. Traditionally it was not possible where file-processing system was used.
- **ACID Properties** − DBMS follows the concepts of **A**tomicity, **C**onsistency, **I**solation, and **D**urability *normally shortened as ACID*. These concepts are applied on transactions, which manipulate data in a database. ACID properties help the database stay healthy in multi-transactional environments and in case of failure.
- **Multiuser and Concurrent Access** − DBMS supports multi-user environment and allows them to access and manipulate data in parallel. Though there are restrictions on transactions when users attempt to handle the same data item, but users are always unaware of them.

- **Multiple views** − DBMS offers multiple views for different users. A user who is in the Sales department will have a different view of database than a person working in the Production department. This feature enables the users to have a concentrate view of the database according to their requirements.
- **Security** − Features like multiple views offer security to some extent where users are unable to access data of other users and departments. DBMS offers methods to impose constraints while entering data into the database and retrieving the same at a later stage. DBMS offers many different levels of security features, which enables multiple users to have different views with different features. For example, a user in the Sales department cannot see the data that belongs to the Purchase department. Additionally, it can also be managed how much data of the Sales department should be displayed to the user. Since a DBMS is not saved on the disk as traditional file systems, it is very hard for miscreants to break the code.

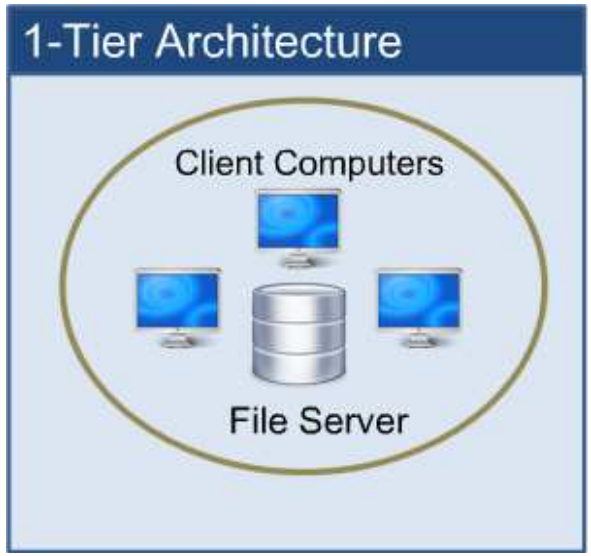
DBMS - ARCHITECTURE

- The design of a DBMS depends on its architecture. It can be centralized or decentralized or hierarchical. The architecture of a DBMS can be seen as either single tier or multi-tier. An n-tier architecture divides the whole system into related but independent **n** modules, which can be independently modified, altered, changed, or replaced.
- In 1-tier architecture, the DBMS is the only entity where the user directly sits on the DBMS and uses it. Any changes done here will directly be done on the DBMS itself. It does not provide handy tools for end-users. Database designers and programmers normally prefer to use single-tier architecture.

- If the architecture of DBMS is 2-tier, then it must have an application through which the DBMS can be accessed. Programmers use 2-tier architecture where they access the DBMS by means of an application. Here the application tier is entirely independent of the database in terms of operation, design, and programming.
- It can be centralized or decentralized or hierarchical. The architecture of a DBMS can be seen as either single tier or multi-tier. The tiers are classified as follows :
  - **1-tier architecture**
  - **2-tier architecture**
  - **3-tier architecture**

1-TIER ARCHITECTURE

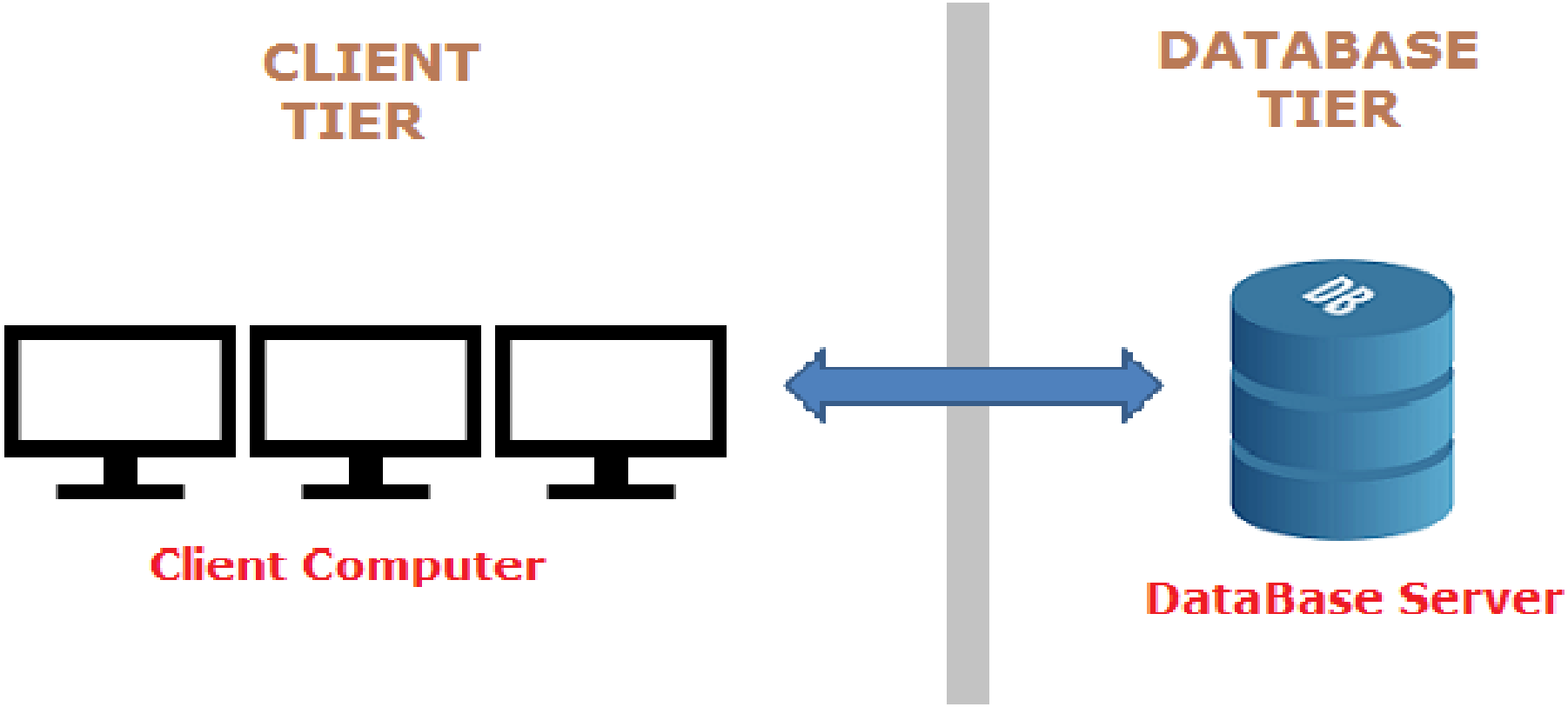
- One-tier architecture involves putting all of the required components for a software application or technology on a single server or platform.
- Basically, a one-tier architecture keeps all of the elements of an application, including the interface, Middleware and back-end data, in one place. Developers see these types of systems as the simplest and most direct way.



2-TIER ARCHITECTURE

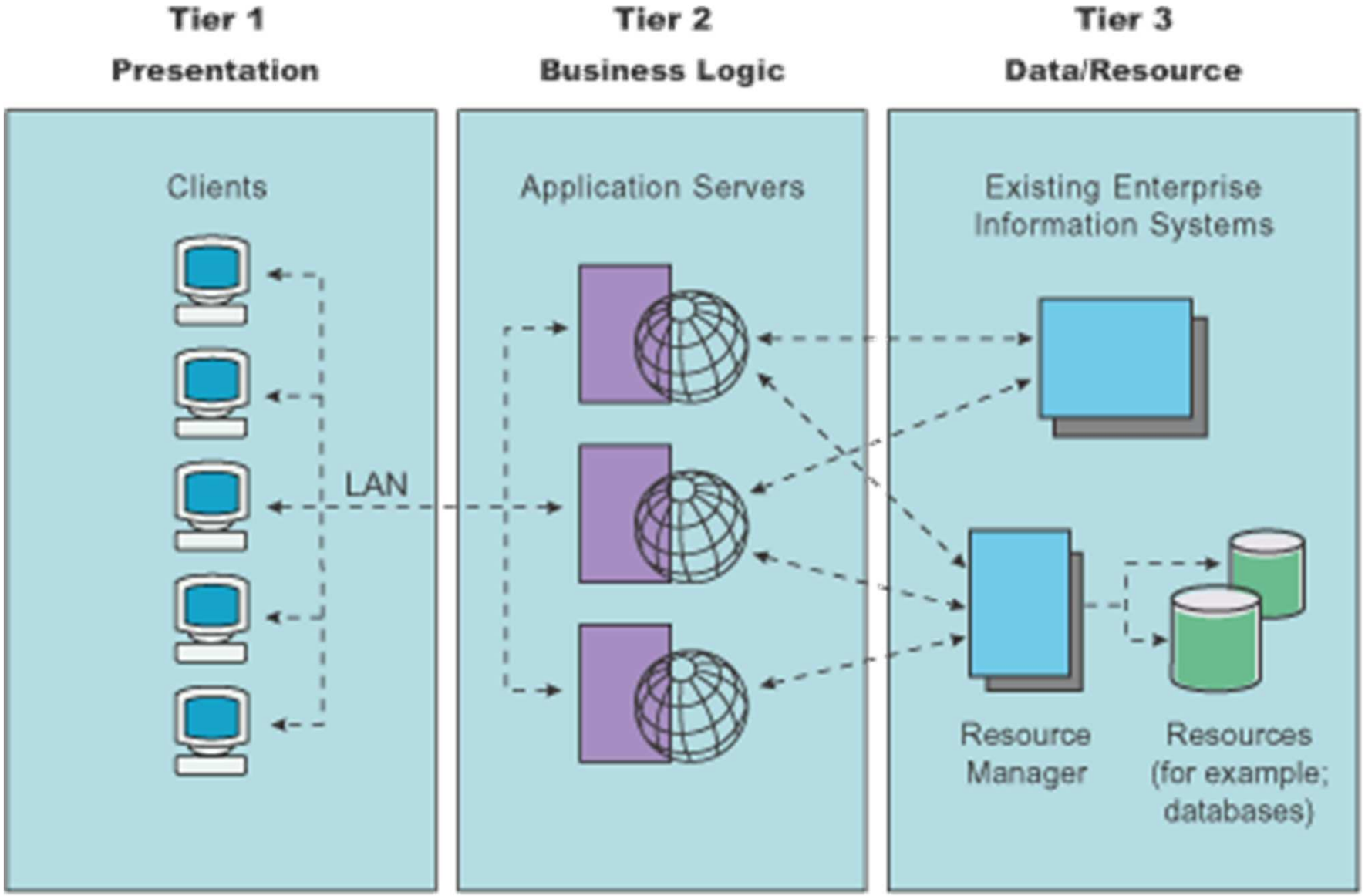
- The two-tier is based on Client Server architecture. 2-tier DBMS architecture includes an **Application layer** between the user and the DBMS, which is responsible to communicate the user's request to the database management system and then send the response from the DBMS to the user.
- The second-tier processes are commonly referred to as the **application logic layer**. These processes manage the **business logic** of the application, and are permitted access to the third-tier services.
- The **application logic layer** is where most of the processing work occurs. Multiple client components can access the second-tier processes simultaneously, so this application logic layer must manage its own transactions.

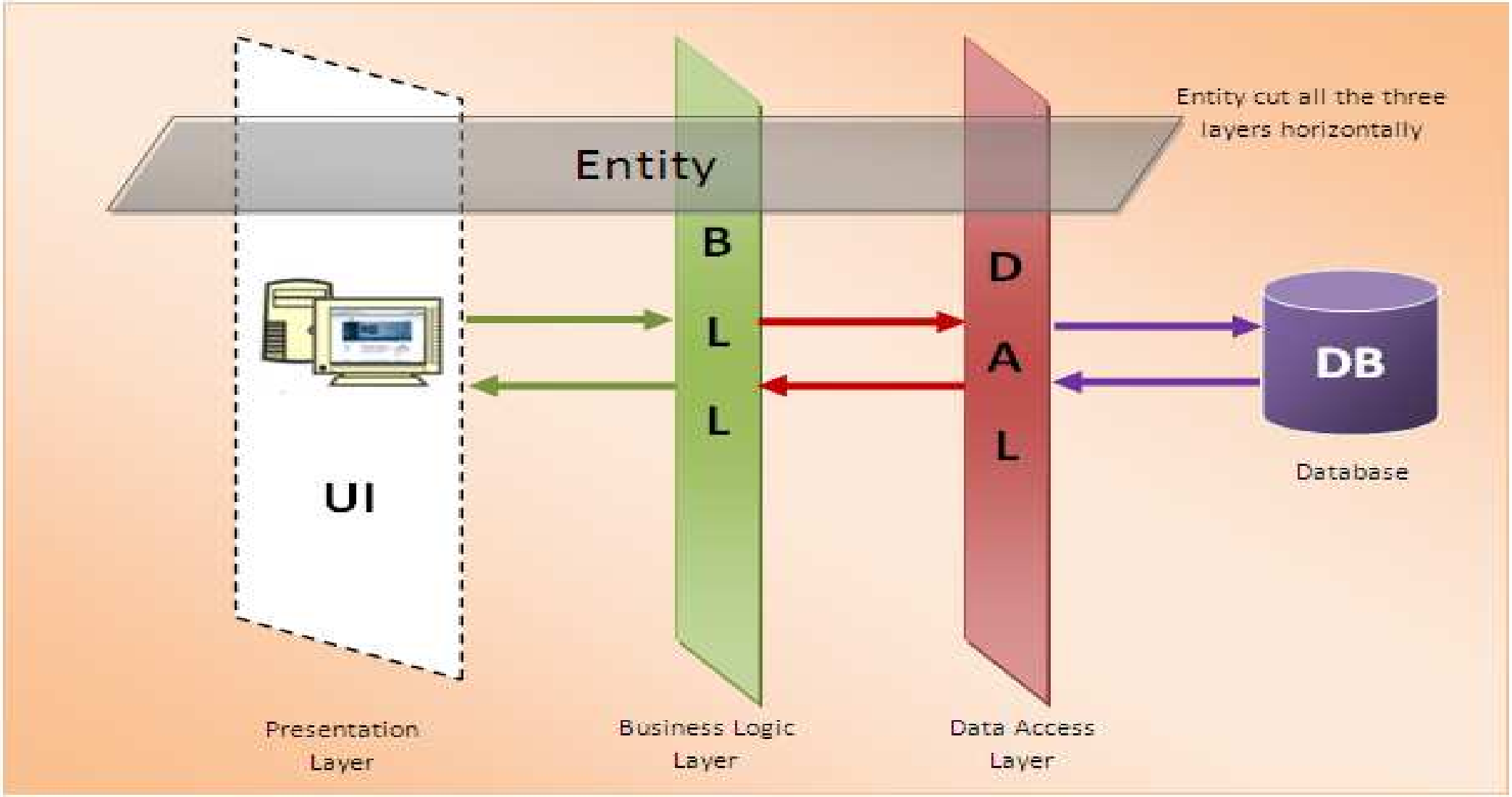
TWO-TIER ARCHITECUTRE



3-TIER ARCHITECTURE

- A 3-tier architecture separates its tiers from each other based on the complexity of the users and how they use the data present in the database. It is the most widely used architecture to design a DBMS.
- This architecture has different usages with different applications. It can be used in web applications and distributed applications. The strength in particular is when using this architecture over distributed systems.


Tier 1
Presentation
Tier 2
Business Logic
Tier 3
Data/Resource
Clients
LAN
Application Servers
Existing Enterprise
Information Systems
Resource
Manager
Resources
(for example;
databases)



[Basic 3-Tire architecture]

- **Database (Data) Tier** − At this tier, the database resides along with its query processing languages. We also have the relations that define the data and their constraints at this level.
- **Application (Middle) Tier** − At this tier reside the application server and the programs that access the database. For a user, this application tier presents an abstracted view of the database. End-users are unaware of any existence of the database beyond the application. At the other end, the database tier is not aware of any other user beyond the application tier. Hence, the application layer sits in the middle and acts as a mediator between the end-user and the database.

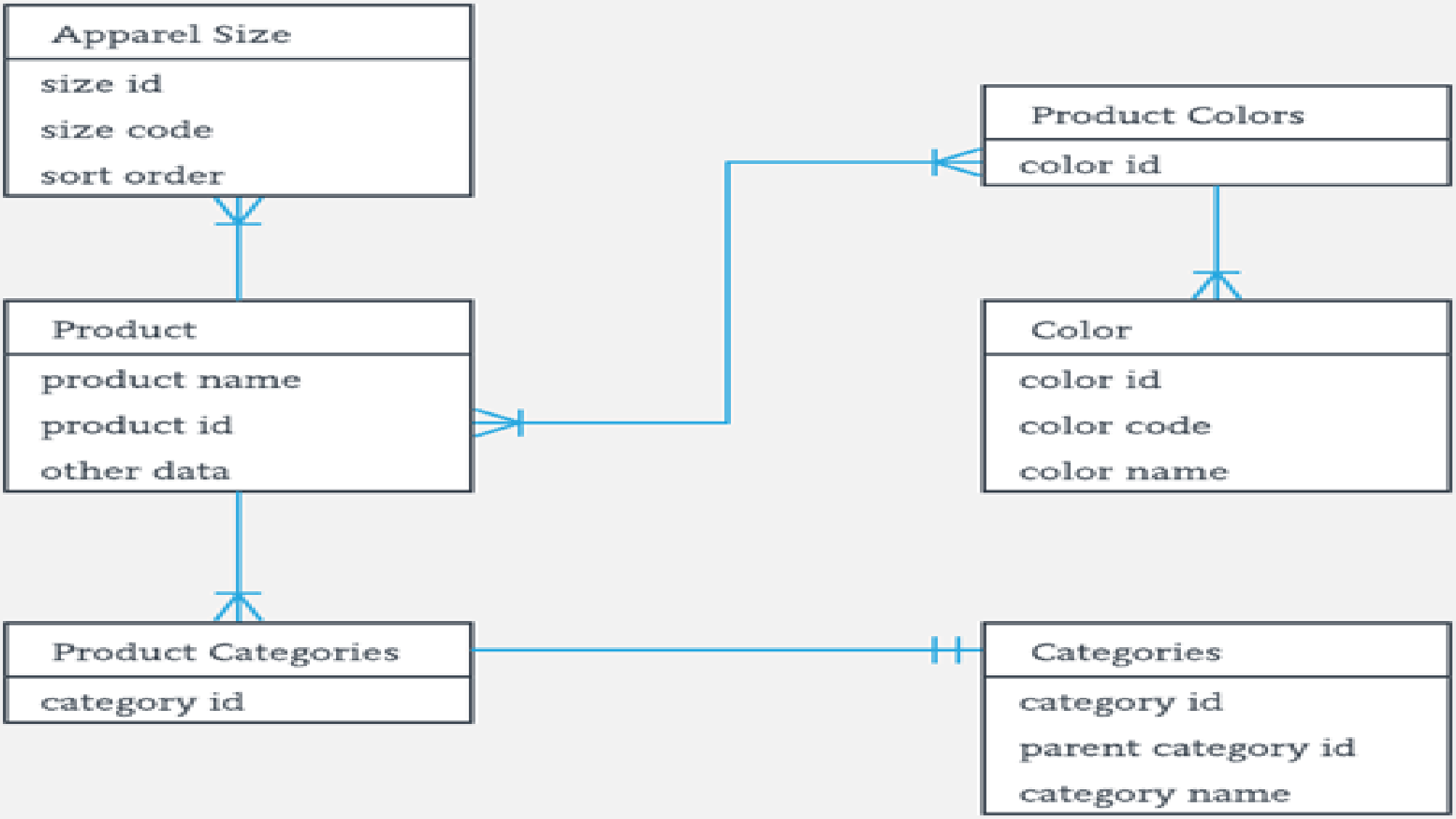
- **User (Presentation) Tier** − End-users operate on this tier and they know nothing about any existence of the database beyond this layer. At this layer, multiple views of the database can be provided by the application. All views are generated by applications that reside in the application tier.

ENTITY- RELATIONSHIP DATA MODEL

- The **ER** or **(Entity Relational Model)** is a high-level conceptual data model diagram. Entity-Relation model is based on the notion of real-world entities and the relationship between them.
- ER modeling helps you to analyze data requirements systematically to produce a well-designed database. So, it is considered a best practice to complete ER modeling before implementing your database.
- An **Entity–relationship model (ER model)** describes the structure of a database with the help of a diagram, which is known as **Entity Relationship Diagram (ER Diagram)**.
- An ER model is a design or blueprint of a database that can later be implemented as a database. The main components of E-R model are: entity set and relationship set.

What is ER Diagrams

- Entity relationship diagram displays the relationships of entity set stored in a database.
- In other words, we can say that ER diagrams help you to explain the logical structure of databases.
- At first look, an ER diagram looks very similar to the flowchart. However, ER Diagram includes many specialized symbols, and its meanings make this model unique.


Apparel Size
size id
size code
sort order
Product Colors
color id
Product
product name
product id
other data
Color
color id
color code
color name
Product Categories
category id
Categories
category id
parent category id
category name

- ER model allows you to draw Database Design
- It is an easy to use graphical tool for modeling data
- Widely used in Database Design
- It is a GUI representation of the logical structure of a Database
- It helps you to identifies the entities which exist in a system and the relationships between those entities.

Why use ER Diagrams

- Helps you to define terms related to entity relationship modeling
- Provide a preview of how all your tables should connect, what fields are going to be on each table
- Helps to describe entities, attributes, relationships
- ER diagrams are translatable into relational tables which allows you to build databases quickly
- ER diagrams can be used by database designers as a blueprint for implementing data in specific software applications
- The database designer gains a better understanding of the information to be contained in the database with the help of ERP diagram
- ERD is allowed you to communicate with the logical structure of the database to users

Components of the ER Diagram

This model is based on three basic concepts:

- Entities
- Attributes
- Relationships

For example, in a University database, we might have entities for Students, Courses, and Lecturers. Students entity can have attributes like Rollno, Name, and DeptID. They might have relationships with Courses and Lecturers.

Entity Name

Entity

Person,place,object,event or concept about which data is to be maintained

Example: Car, Student

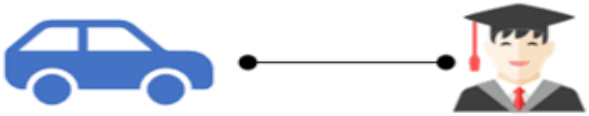
Jack

Attribute Name

Attribute

Property or characteristic of an entity

Example: Color of car Entity Name of Student Entity



Relation

Verb Phrase

Association between the instances of one or more entity types

Example: Blue Car Belongs to Student Jack

WHAT IS ENTITY

- A real-world thing either living or non-living that is easily recognizable and non-recognizable. It is anything in the enterprise that is to be represented in our database. It may be a physical thing or simply a fact about the enterprise or an event that happens in the real world.
- An entity can be place, person, object, event or a concept, which stores data in the database. The characteristics of entities are must have an attribute, and a unique key. Every entity is made up of some 'attributes' which represent that entity.

Examples of entities:

- **Person:** Employee, Student, Patient
- **Place:** Store, Building
- **Object:** Machine, product, and Car
- **Event:** Sale, Registration, Renewal
- **Concept:** Account, Course

Entity set: (Notation of an Entity)

Student

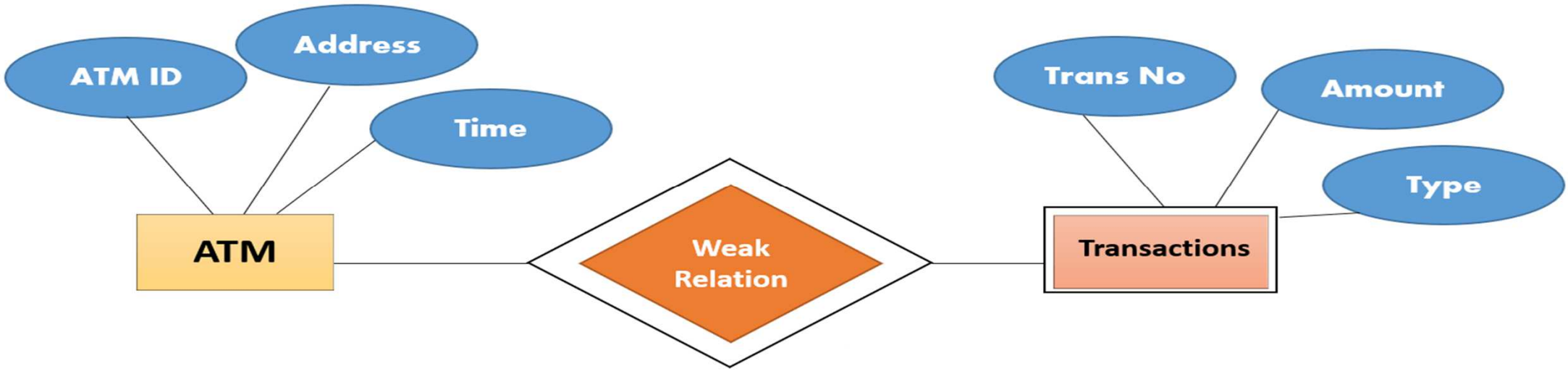
- An entity set is a group of similar kind of entities. It may contain entities with attribute sharing similar values. Entities are represented by their properties, which also called attributes. All attributes have their separate values. For example, a student entity may have a name, age, class, as attributes.

Example of Entities:

- A university may have some departments. All these departments employ various lecturers and offer several programs.
- Some courses make up each program. Students register in a particular program and enroll in various courses. A lecturer from the specific department takes each course, and each lecturer teaches a various group of students.

Weak Entities

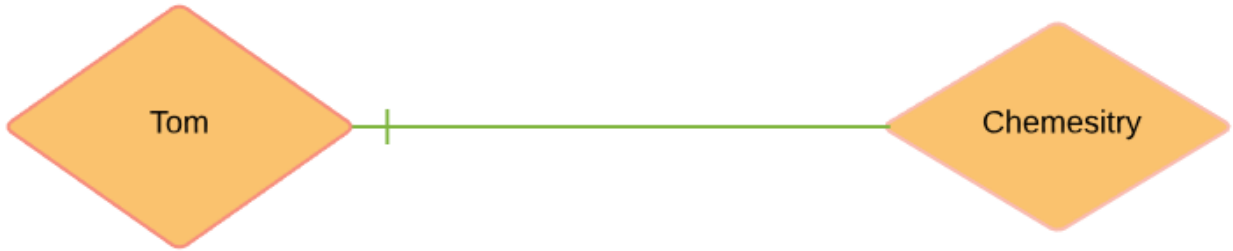
- A weak entity is a type of entity which doesn't have its key attribute. It can be identified uniquely by considering the primary key of another entity. For that, weak entity sets need to have participation.
- In below example, "Trans No" is a discriminator within a group of transactions in an ATM.



STRONG ENTITY SET	WEAK ENTITY SET
Strong entity set always has a primary key.	It does not have enough attributes to build a primary key.
It is represented by a rectangle symbol.	It is represented by a double rectangle symbol.
It contains a Primary key represented by the underline symbol.	It contains a Partial Key which is represented by a dashed underline symbol.
The member of a strong entity set is called as dominant entity set.	The member of a weak entity set called as a subordinate entity set.
Primary Key is one of its attributes which helps to identify its member.	In a weak entity set, it is a combination of primary key and partial key of the strong entity set.
In the ER diagram the relationship between two strong entity set shown by using a diamond symbol.	The relationship between one strong and a weak entity set shown by using the double diamond symbol.
The connecting line of the strong entity set with the relationship is single.	The line connecting the weak entity set for identifying relationship is double.

Relationship

- Relationship is nothing but an association among two or more entities. E.g., Tom works in the Chemistry department.
- A relationship captures how entities are related to one another. Relationships can be thought of as verbs, linking two or more nouns.
- Examples: an *owns* relationship between a company and a computer, a *supervises* relationship between an employee and a department, a *performs* relationship between an artist and a song, a *proves* relationship between a mathematician and a conjecture, etc.



- Entities take part in relationships. We can often identify relationships with verbs or verb phrases.

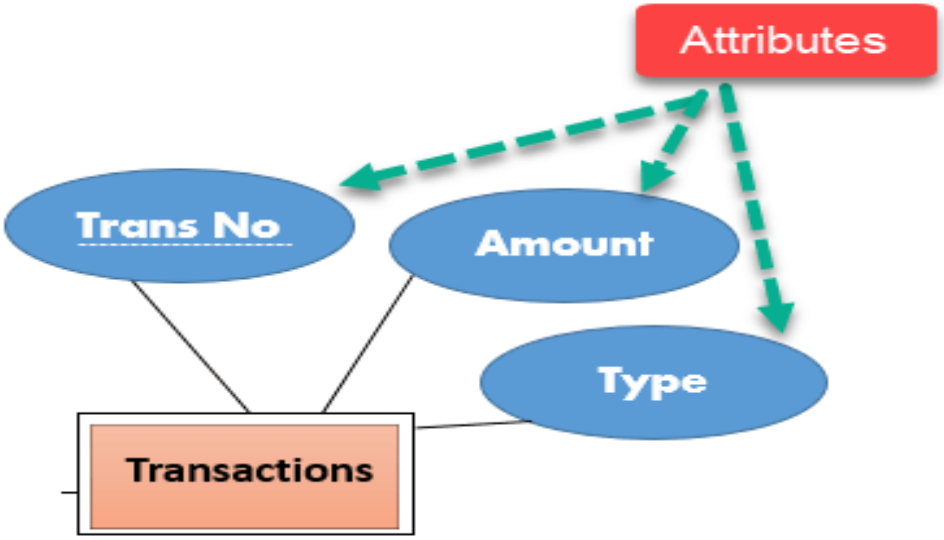
For example:

- You are attending this lecture
- I am giving the lecture
- Just look entities, we can classify relationships according to relationship-types:
  - A student attends a lecture
  - A lecturer is giving a lecture.

ATTRIBUTES

- It is a single-valued property of either an entity-type or a relationship-type. An attribute is represented by an Ellipse. **For example**, a lecture might have attributes: time, date, duration, place, etc.
- OR An attribute is a property, trait, or characteristic of an entity, relationship, or another attribute.

For example, a Transactions might have attributes: Trans No, Amount, Type etc.



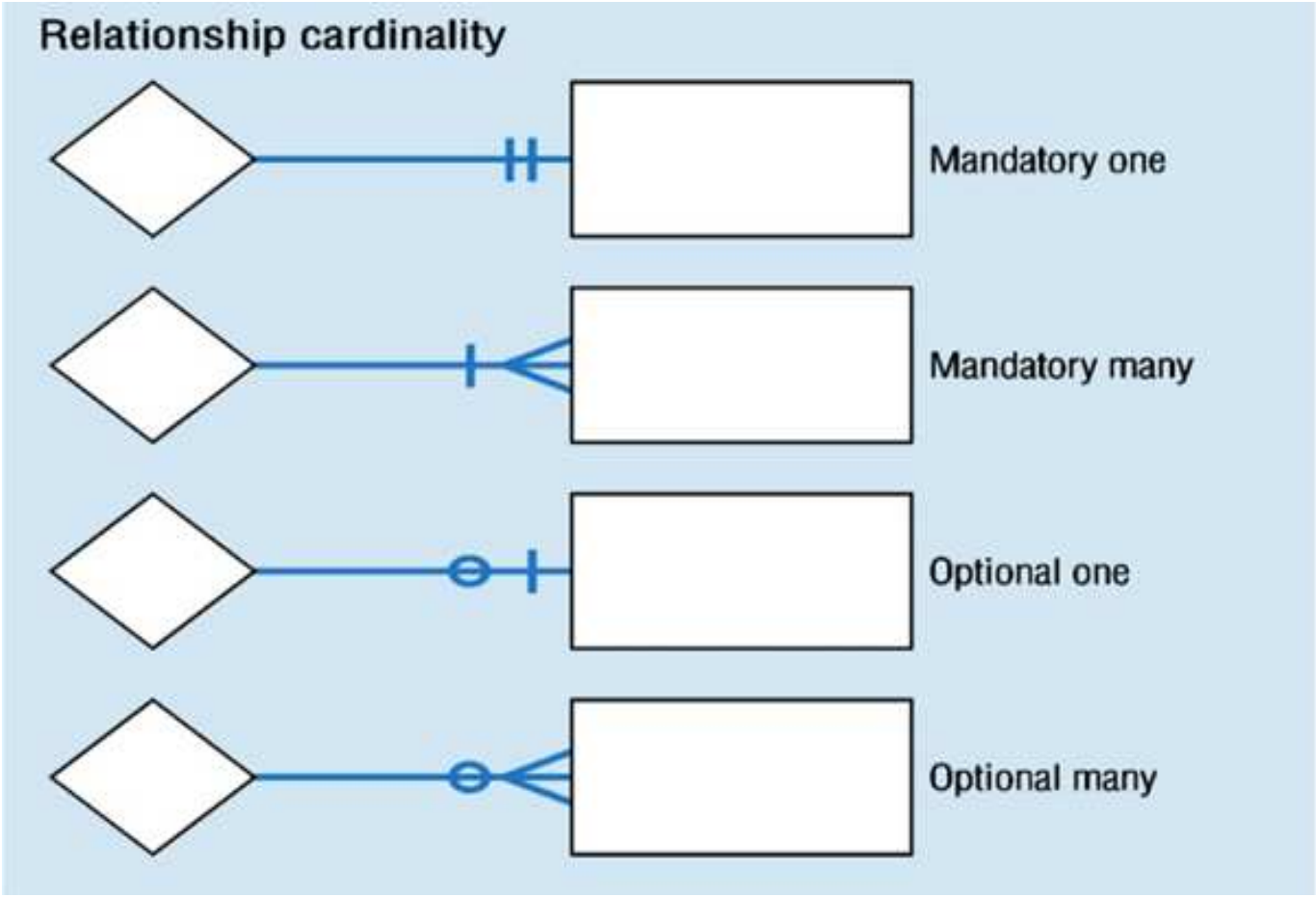
TYPES OF ATTRIBUTES	DESCRIPTION
Simple attribute	Simple attributes can't be divided any further. For example, a student's contact number. It is also called an atomic value.
Composite attribute	It is possible to break down composite attribute. For example, a student's full name may be further divided into first name, second name, and last name.
Derived attribute	This type of attribute does not include in the physical database. However, their values are derived from other attributes present in the database. For example, age should not be stored directly. Instead, it should be derived from the DOB of that employee.
Multivalued attribute	Multivalued attributes can have more than one values. For example, a student can have more than one mobile number, email address, etc.

Cardinality

- Defines the numerical attributes of the relationship between two entities or entity sets.
- **Cardinality** specifies how many instances of an entity relate to one instance of another entity. Ordinality is also closely linked to cardinality. While cardinality specifies the occurrences of a relationship, ordinality describes the relationship as either mandatory or optional. In other words, cardinality specifies the maximum number of relationships and ordinality specifies the absolute minimum number of relationships.

Different types of cardinal relationships are:

- One-to-One Relationships
- One-to-Many Relationships
- Many to One Relationships
- Many-to-Many Relationships

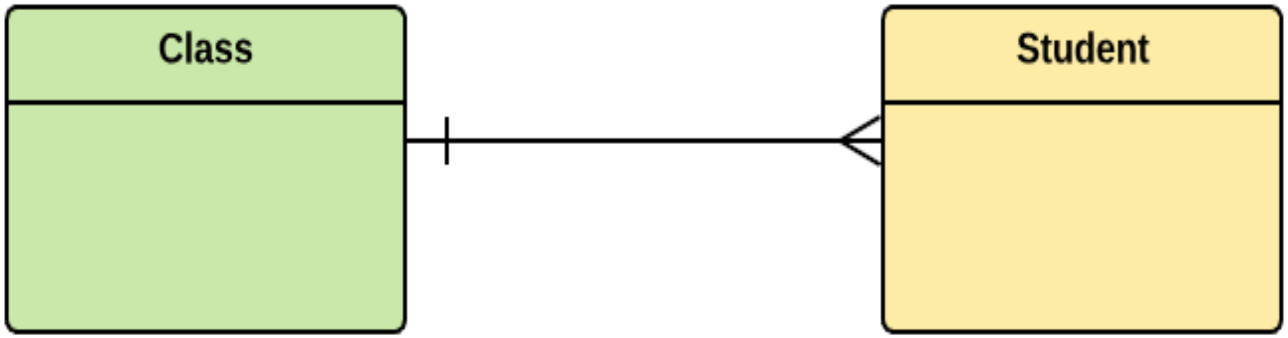


One-to-one

- One entity from entity set X can be associated with at most one entity of entity set Y and vice versa.
- **For Example**: One student can register for numerous courses. However, all those courses have a single line back to that one student.

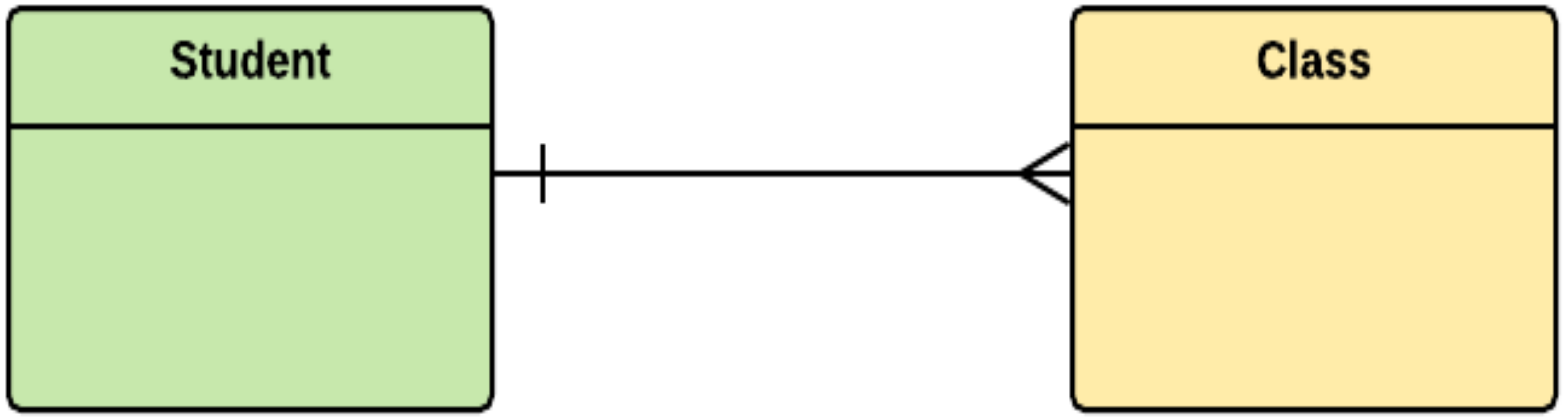
One-to-many

- One entity from entity set X can be associated with multiple entities of entity set Y, but an entity from entity set Y can be associated with at least one entity.
- **For example**, one class is consisting of multiple students.



Many to One

- More than one entity from entity set X can be associated with at most one entity of entity set Y. However, an entity from entity set X may or may not be associated with more than one entity from entity set X.
- **For example**, many students belong to the same class.



Many to Many

- One entity from X can be associated with more than one entity from Y and vice versa.
- **For example**, Students as a group are associated with multiple faculty members, and faculty members can be associated with multiple students.

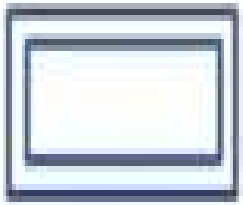
ER- Diagram Notations

R- Diagram is a visual representation of data that describe how data is related to each other.

- **Rectangles:** This symbol represent entity types
- **Ellipses :** Symbol represent attributes
- **Diamonds:** This symbol represents relationship types
- **Lines:** It links attributes to entity types and entity types with other relationship types
- **Primary key:** attributes are underlined
- **Double Ellipses:** Represent multi-valued attributes

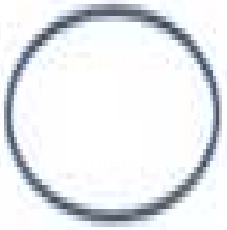
Entity or Strong Entity

Relationship

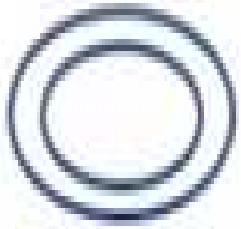


Weak Entity

Weak Relationship



Attribute



Multivalued Attribute

Steps to Create an ER Diagram

- **Step 1) Entity Identification**
- **Step 2) Relationship Identification**
- **Step 3) Cardinality Identification**
- **Step 4) Identify Attributes**
- **Step 5) Create the ERD**

Create ER Diagram

- 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