

DATABASE MANAGEMENT SYSTEMS

(MySQL)

CHAPTER - 1

INTRODUCTION TO SQL

What is Data?

- ✓ Data is a raw and unorganized fact that required to be processed to make it meaningful.
- ✓ Data can be simple at the same time unorganized unless it is organized.
- ✓ Generally, data comprises facts, observations, perceptions, numbers, characters, symbols, image, etc.

What is Information?

- ✓ Information is a set of data which is processed in a meaningful way according to the given requirement.
- ✓ Information is processed, structured, or presented in a given context to make it meaningful and useful.
- ✓ It is processed data which includes data that possess context, relevance, and purpose. It also involves manipulation of raw data.

Difference between Data and Information?

Data	Information
Data is unorganized and unrefined facts	Information comprises processed, organized data presented in a meaningful context
Data is an individual unit that contains raw materials which do not carry any specific meaning.	Information is a group of data that collectively carries a logical meaning.
Data doesn't depend on information.	Information depends on data.
Raw data alone is insufficient for decision making	Information is sufficient for decision making
An example of data is a student's test score	The average score of a class is the information derived from the given data.

What is Database?

- ✓ A database is a vast collection of data that is stored and retrieved electronically from a system.
- ✓ This structured data stored in the database is processed, manipulated, controlled and updated to perform various operations.
- ✓ For example: The college Database organizes the data about the admin, staff, students and faculty etc.
- ✓ Using the database, you can easily retrieve, insert, and delete the information.

What is the need of a Database?

- ✓ Manage vast volumes of data.
- ✓ Manipulate and update data.
- ✓ Keeps your data safe and secure.

What is Database Management System (DBMS) ?

- ✓ Database management system is a software which is used to manage the database.
- ✓ For example: SQL, MySQL, Oracle, etc are a very popular commercial database which is used in different applications.
- ✓ DBMS provides an interface to perform various operations like database creation, storing data in it, updating data, creating a table in the database and a lot more.
- ✓ It provides protection and security to the database. In the case of multiple users, it also maintains data consistency.

DBMS allows users the following tasks:

- ✓ **Data Definition:** It is used for creation, modification, and removal of definition that defines the organization of data in the database.
- ✓ **Data Updation:** It is used for the insertion, modification, and deletion of the actual data in the database.
- ✓ **Data Retrieval:** It is used to retrieve the data from the database which can be used by applications for various purposes.
- ✓ **User Administration:** It is used for registering and monitoring users, maintain data integrity, enforcing data security, dealing with concurrency control, monitoring performance and recovering information corrupted by unexpected failure.

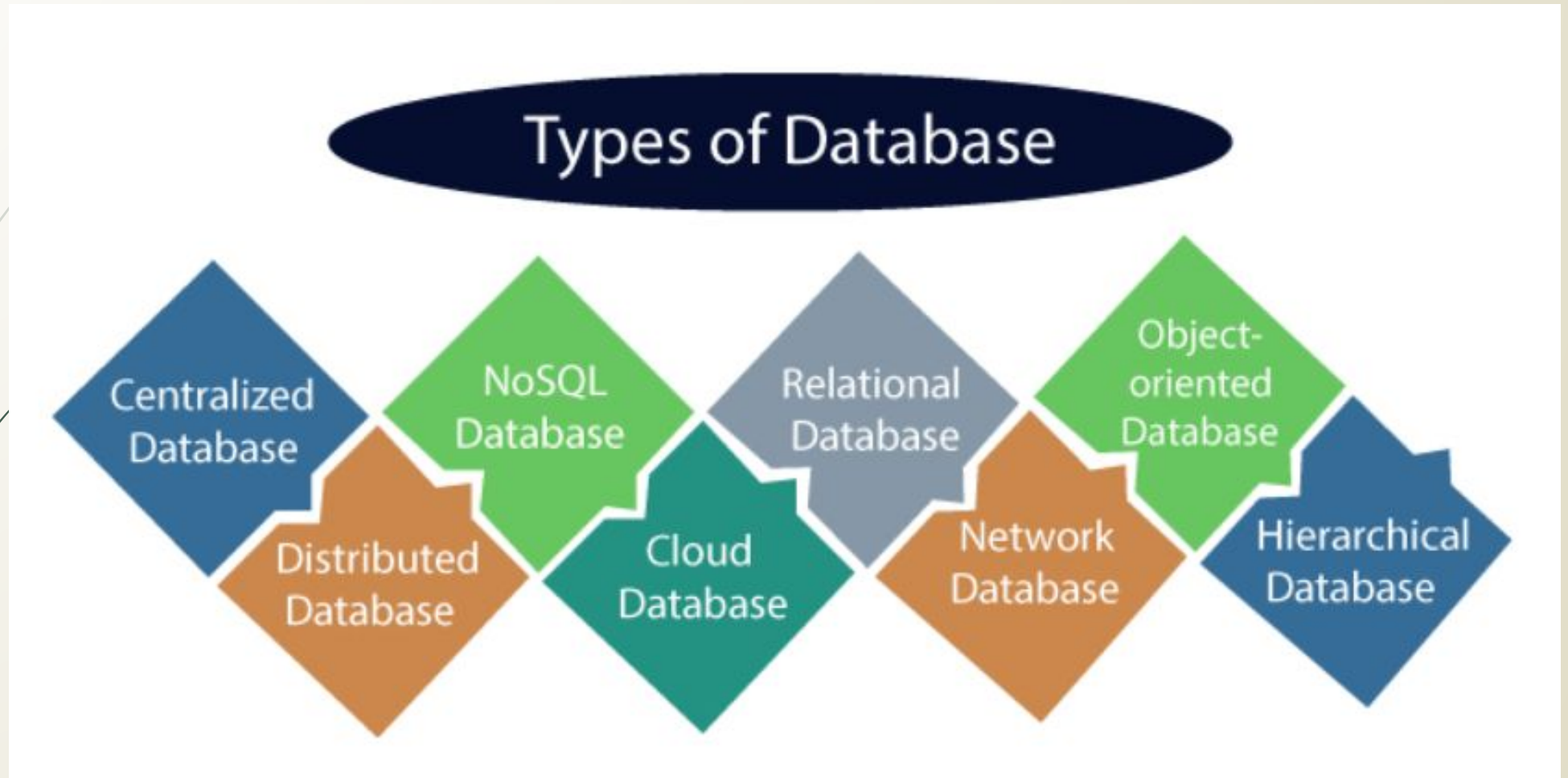
Advantages of DBMS:

- ✓ **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.
- ✓ **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 electric failure or database corruption then the data may be lost forever.

Types of Database:



Centralized Database:

- ✓ This type of database that stores data at a centralized database system.
- ✓ It comforts the users to access the stored data from different locations through several applications.
- ✓ These applications contain the authentication process to let users access data securely.
- ✓ An example of a Centralized database can be Central Library that carries a central database of each library in a college/university.

Distributed Database:

- ✓ Unlike a centralized database system, in distributed systems, data is distributed among different database systems of an organization.
- ✓ These database systems are connected via communication links. Such links help the end-users to access the data easily.
- ✓ Examples of the Distributed database are Apache Cassandra, HBase, Ignite, etc.

We can further divide a distributed database system into:

- ✓ **Homogeneous DDB:** Those database systems which execute on the same operating system and use the same application process and carry the same hardware devices.
- ✓ **Heterogeneous DDB:** Those database systems which execute on different operating systems under different application procedures, and carries different hardware devices.

Relational Database:

- ✓ This database is based on the relational data model, which stores data in the form of rows(tuple) and columns(attributes), and together forms a table(relation).
- ✓ A relational database uses SQL for storing, manipulating, as well as maintaining the data.
- ✓ E.F. Codd invented the database in 1970. Each table in the database carries a key that makes the data unique from others.
- ✓ Examples of Relational databases are MySQL, Microsoft SQL Server, Oracle, etc.

NoSQL Database:

- ✓ Non-SQL/Not Only SQL is a type of database that is used for storing a wide range of data sets.
- ✓ It is not a relational database as it stores data not only in tabular form but in several different ways.
- ✓ It came into existence when the demand for building modern applications increased.
- ✓ Thus, NoSQL presented a wide variety of database technologies in response to the demands.

We can further divide a NoSQL database into the following four types:

- ✓ **Key-value storage:** It is the simplest type of database storage where it stores every single item as a key (or attribute name) holding its value, together.
- ✓ **Document-oriented Database:** A type of database used to store data as JSON-like document. It helps developers in storing data by using the same document-model format as used in the application code.
- ✓ **Graph Databases:** It is used for storing vast amounts of data in a graph-like structure. Most commonly, social networking websites use the graph database.
- ✓ **Wide-column stores:** It is similar to the data represented in relational databases. Here, data is stored in large columns together, instead of storing in rows.

Cloud Database:

- ✓ A type of database where data is stored in a virtual environment and executes over the cloud computing platform.
- ✓ It provides users with various cloud computing services (SaaS, PaaS, IaaS, etc.) for accessing the database.

There are numerous cloud platforms, but the best options are:

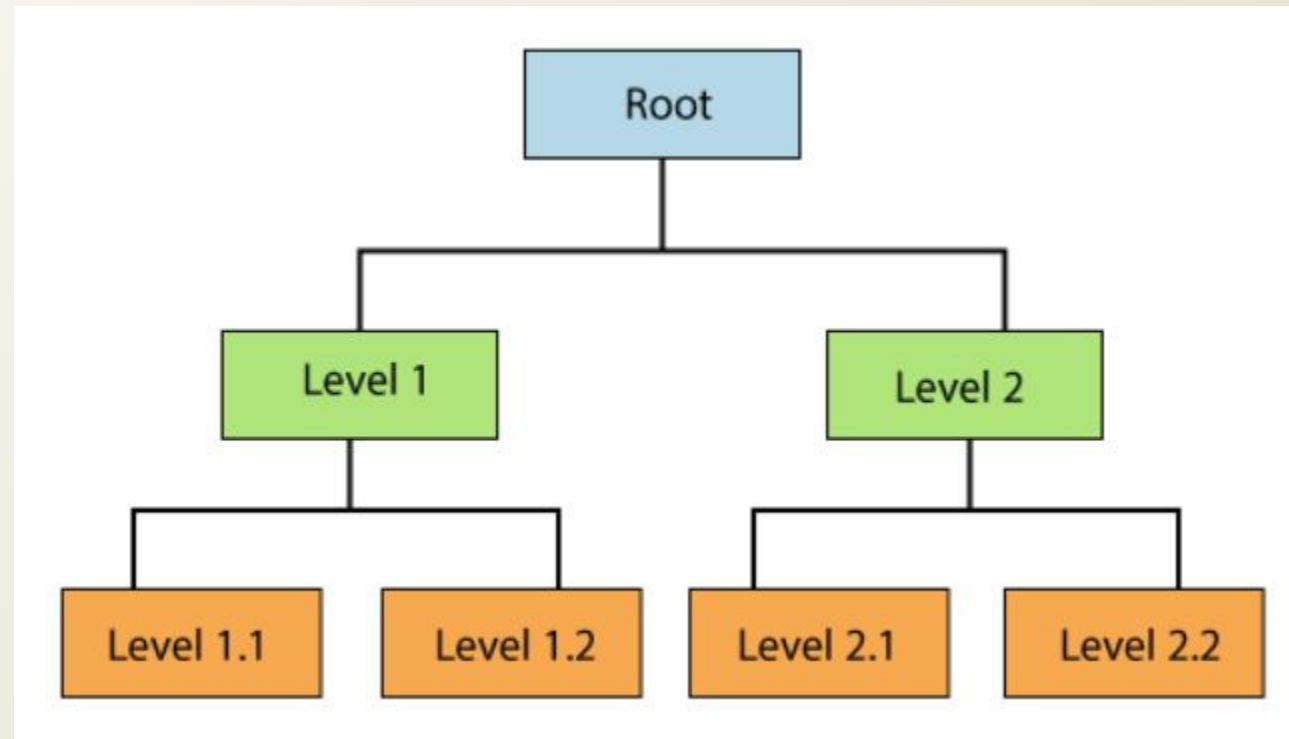
- ✓ Amazon Web Services(AWS)
- ✓ Microsoft Azure
- ✓ Google Cloud Storage, etc.

Object-oriented Databases:

- ✓ The type of database that uses the object-based data model approach for storing data in the database system.
- ✓ The data is represented and stored as objects which are similar to the objects used in the object-oriented programming language.

Hierarchical Databases:

- ✓ It is the type of database that stores data in the form of parent-children relationship nodes. Here, it organizes data in a tree-like structure.
- ✓ Data get stored in the form of records that are connected via links.
- ✓ Each child record in the tree will contain only one parent. On the other hand, each parent record can have multiple child records.



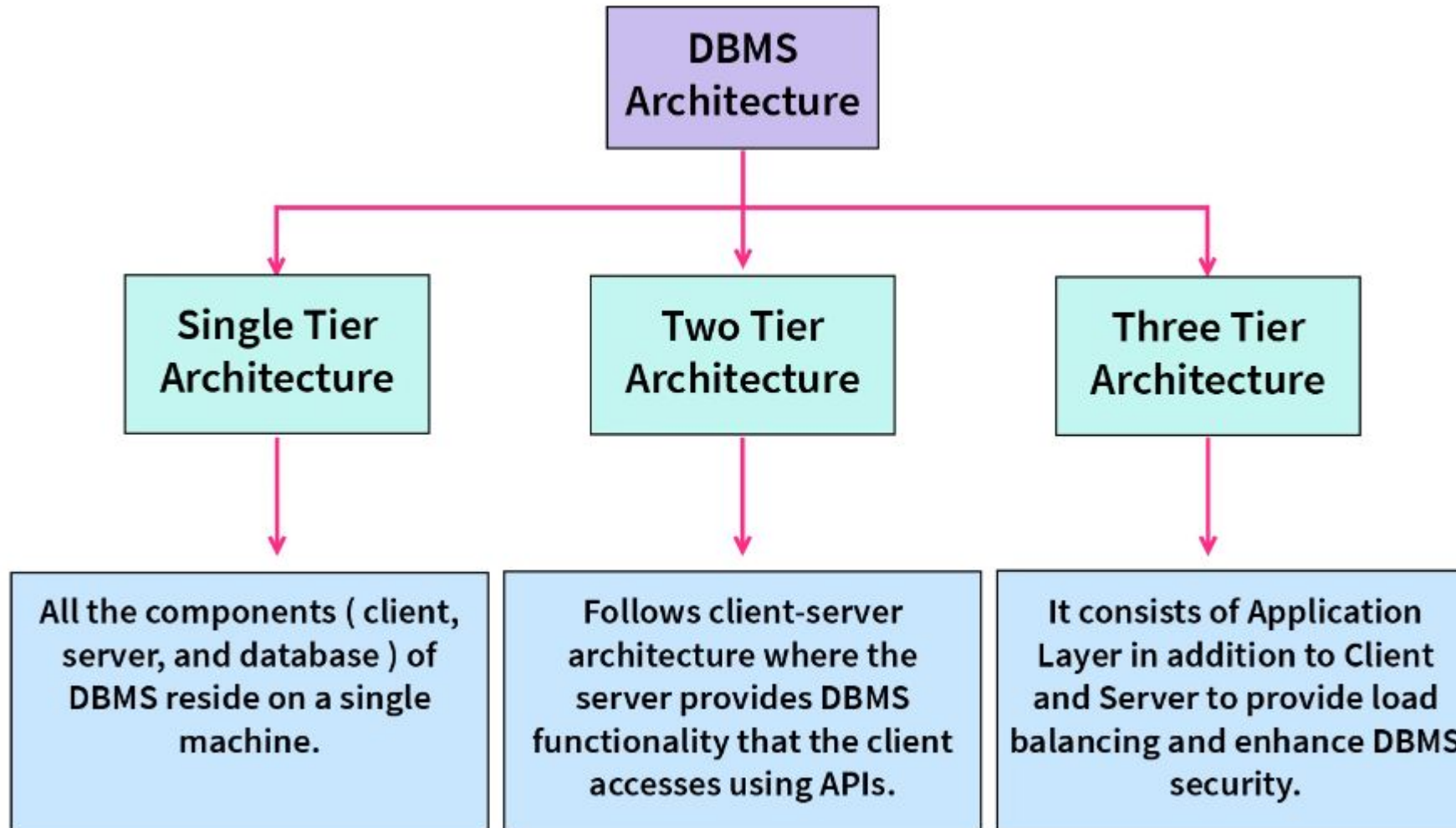
Network Databases:

- ✓ It is the database that typically follows the network data model.
- ✓ Here, the representation of data is in the form of nodes connected via links between them.
- ✓ Unlike the hierarchical database, it allows each record to have multiple children and parent nodes to form a generalized graph structure.

DBMS Architecture

- ✓ A Database Architecture is a representation of DBMS design.
- ✓ It helps to design, develop, implement, and maintain the database management system.
- ✓ A DBMS architecture allows dividing the database system into individual components that can be independently modified, changed, replaced, and altered.
- ✓ It also helps to understand the components of a database.
- ✓ A Database stores critical information and helps access data quickly and securely. Therefore, selecting the correct Architecture of DBMS helps in easy and efficient data management.

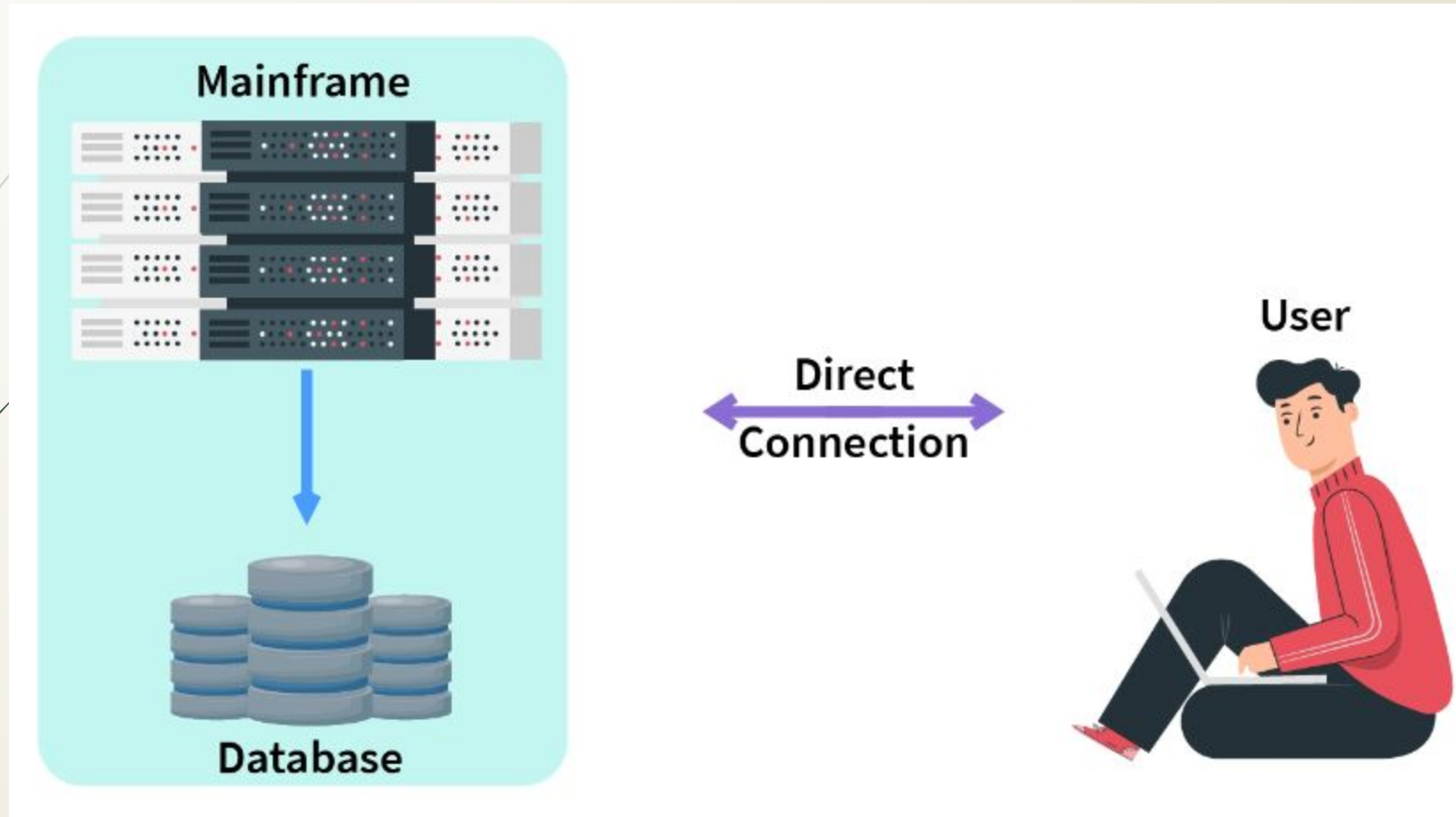
Types of DBMS Architecture



Single Tier Architecture

- ✓ Single Tier DBMS Architecture is the most straightforward DBMS architecture.
- ✓ All the DBMS components reside on a single server or platform, i.e., the database is directly accessible by the end-user.
- ✓ Because of this direct connection, the DBMS provides a rapid response, due to which programmers widely use this architecture to enhance the local application.
- ✓ In this structure, any modifications done by the client are reflected directly in the database, and all the processing is done on a single server.
- ✓ Also, no network connection is required to perform actions on the database.
- ✓ This database management system is also known as the local database system.

Single Tier Architecture



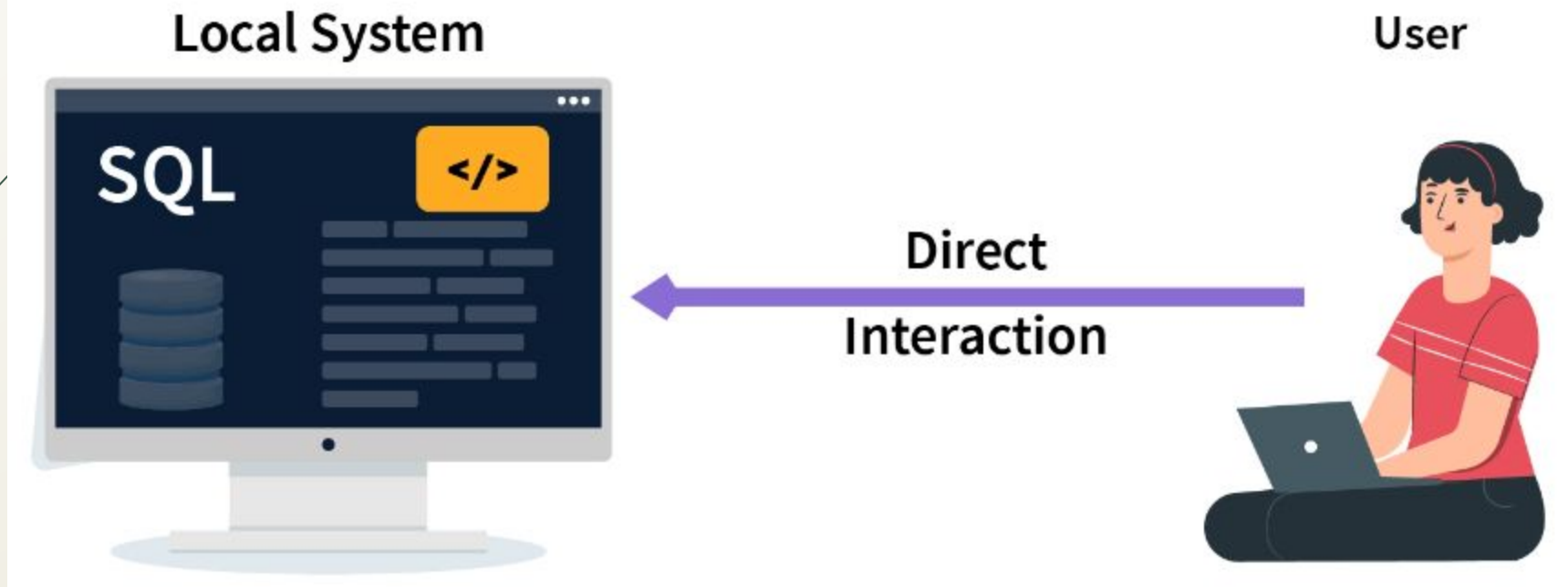
Single Tier DBMS Architecture is used whenever:

- ✓ The data isn't changed frequently.
- ✓ No multiple users are accessing the database system.
- ✓ We need a direct and simple way to modify or access the database for application development.

Example of Single Tier DBMS Architecture:

- ✓ In order to learn the Structure Query Language (SQL), we set up our SQL server and the database on our local system.
- ✓ This SQL server enables us to directly interact with the relational database and execute certain operations without requiring any network connection.
- ✓ This whole setup to learn SQL queries is an example of Single-Tier DBMS architecture.

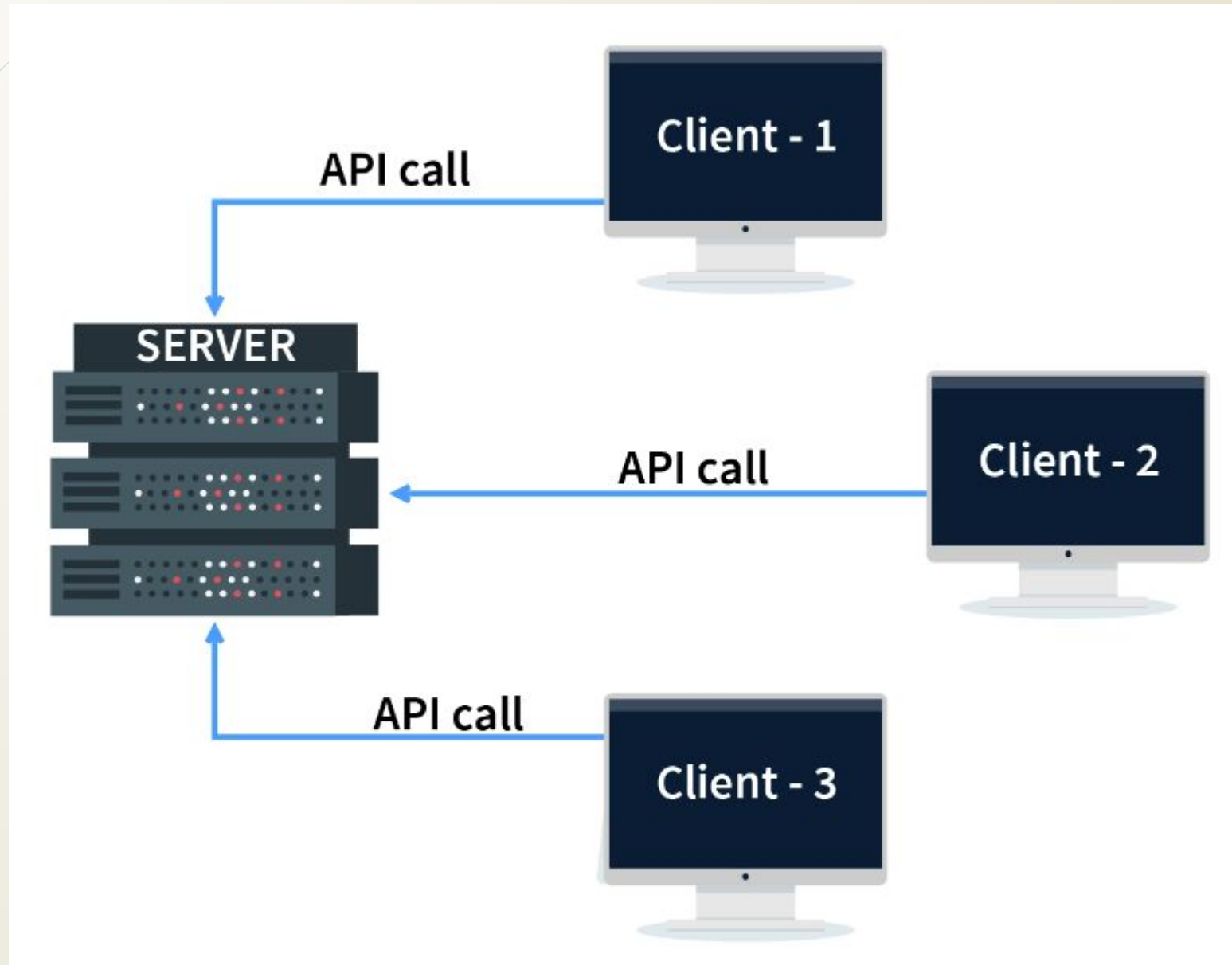
Example of Single Tier DBMS Architecture



Two Tier Architecture

- ✓ Two-Tier DBMS Architecture is similar to the fundamental concept of client-server architecture.
- ✓ In a two-tier structure, the server provides the database functionality and it allows the clients to perform operations on the database through a direct internet connection via APIs (Application Programming Interface), for example:
- ✓ ODBC: Open Database Connectivity.
- ✓ JDBC: Java Database Connectivity.
- ✓ The Two-Tier DBMS Architecture is used when we wish to access the DBMS with the help of an application.
- ✓ Client-side applications can access the database server directly with the help of API calls, making the application independent of the database in terms of operation, design, and programming.

Two Tier Architecture

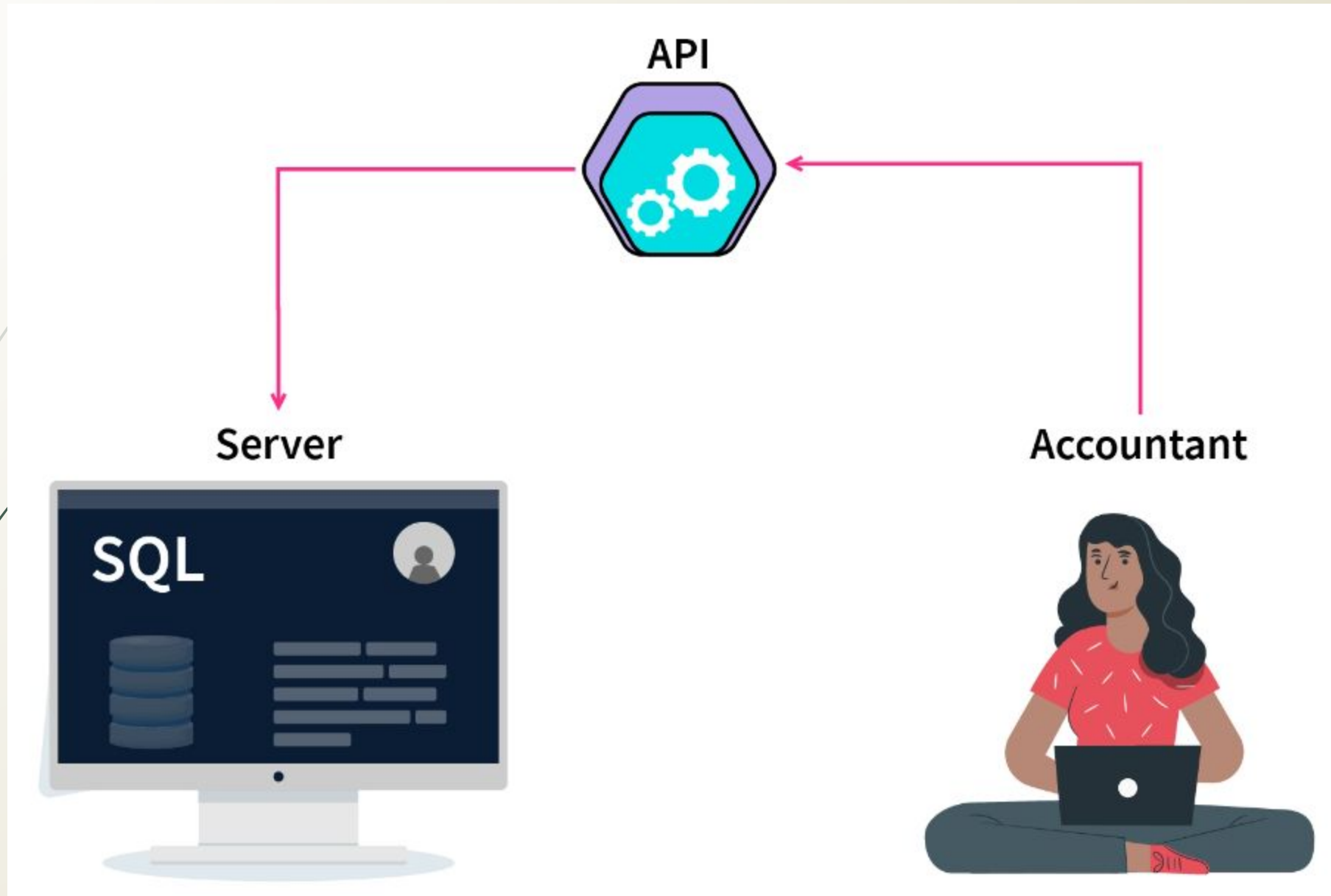


The main advantages of having a two-tier architecture over a single tier are:

- ✓ Multiple users can use it at the same time. Hence, it can be used in an organization.
- ✓ It has high processing ability as the database functionality is handled by the server alone.
- ✓ Faster access to the database due to the direct connection and improved performance.
- ✓ Because of the two independent layers, it's easier to maintain.

Example of Two Tier DBMS Architecture:

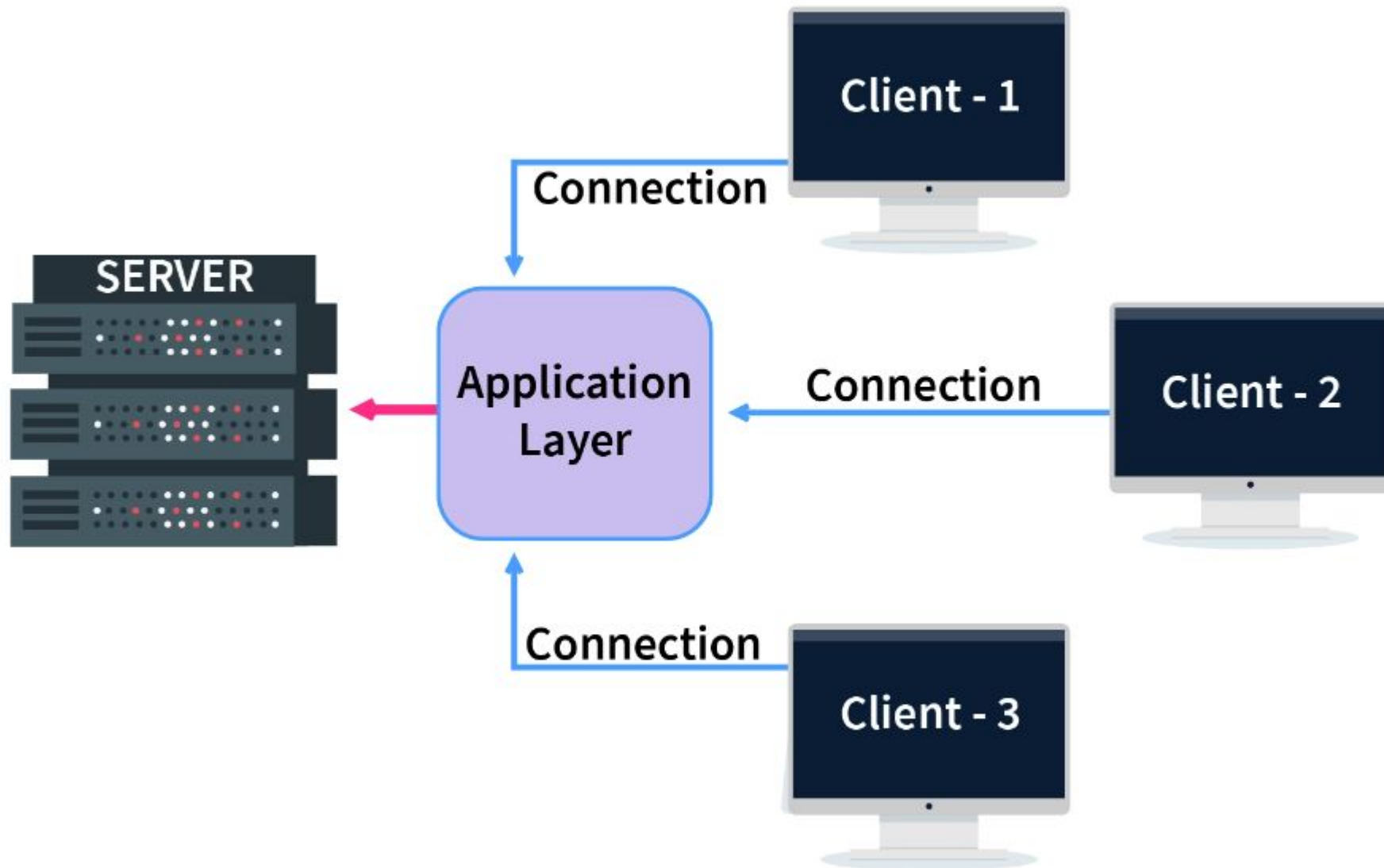
- ✓ Consider a situation where you went to a bank to withdraw some cash.
- ✓ After entering the withdrawal amount and the account details on the withdrawal slip, the banker will go through the server-side database via his credential (API call) and will check whether there is enough balance present or not.
- ✓ This client-server model is an example of Two-Tier DBMS architecture.



Three Tier Architecture

- ✓ Three-Tier DBMS Architecture is the most widely used DBMS architecture in which another layer known as Intermediate or Application layer is added between the server (Database Layer) and the client (Presentation Layer) systems to reduce the query processing load of the server.
- ✓ This application or intermediate layer provides the end-user an abstract view of the database.
- ✓ Since there is no direct connection between the client and the Server, all the user requests are handled by the Application Layer,
- ✓ i.e., the requests sent by the users are checked and verified by the Intermediate Layer before transferring them to the server.
- ✓ This reduces the query processing load from the server and enhances the security of the overall DBMS design as the client can't communicate directly with the database server.
- ✓ Hence, the application layer is responsible for load balancing, query request correctness, and security in Three-Tier DBMS Architecture.

Three Tier Architecture



The main advantages of Three Tier DBMS Architecture are:

- ✓ Scalability - Since the database server isn't aware of any users beyond the application layer and the application layer implements load balancing, there can be as many clients as you want.
- ✓ Data Integrity - Data corruption and bad requests can be avoided because of the checks performed in the application layer on each client request.
- ✓ Security - The removal of the direct connection between the client and server systems via abstraction reduces unauthorized access to the database.

Note - In Three Tier DBMS Architecture, an additional layer (Application Layer) is added between the Client and the Server.

This increases the number of layers present between the DBMS and the end-users, making the implementation of the DBMS structure complex and difficult to maintain.

What is Data Modeling?

- ✓ The process of creating a model for the storage of data in a database is termed as data modeling.
- ✓ It is a theoretical presentation of data objects and associations among various data objects.
- ✓ Data modeling is a process of formulating data in an information system in a structured format.
- ✓ It helps in analyzing data easily which will further help in meeting business requirements.

Why use a data model?

Primary reasons for using a data model are listed below:

- ✓ Visual representation of data helps improve data analysis. It provides a holistic picture of the data which can be used by developers to create a physical database.
- ✓ All important data of an enterprise are accurately presented in the model. The data model reduces the chances of data omission. Data omission can lead to incorrect results and faulty reports.
- ✓ The data model portrays a better understanding of business requirements.
- ✓ It helps in the creation of a robust design that brings the entire data of an organization on the same platform. It assists in identifying the redundant, duplicate, and missing data as well.
- ✓ A qualified data model helps in providing better consistency across all projects of an enterprise.
- ✓ It improves data quality.
- ✓ It helps Project Managers with a better scope and quality management. It also improves performance to the core.
- ✓ It defines relational tables, stored procedures, and primary and foreign keys.

Three categories of Data Model:

- ✓ Data models are used to define the data and how it is stored in a database and to set a relation between data elements.

- ✓ Conceptual Model

This level defines what needs to be present in the structure of the model in order to define and organize business concepts. It mainly focuses on business-oriented entries, attributes, and relations. It is basically designed by Data Architects and Business Stakeholders.

- ✓ Logical Model

The logical model defines how the model should be implemented. It broadly includes all kinds of data that need to be captured such as tables, columns, etc. This model is generally designed by Business Analysts and Data Architects.

- ✓ Physical Model

The physical model defines how to implement a data model with the help of the database management system. It outlines the implementation methodology in terms of tables, CRUD operations, indexes, partitioning, etc. It is created by Database Administrators and Developers.

ER 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.

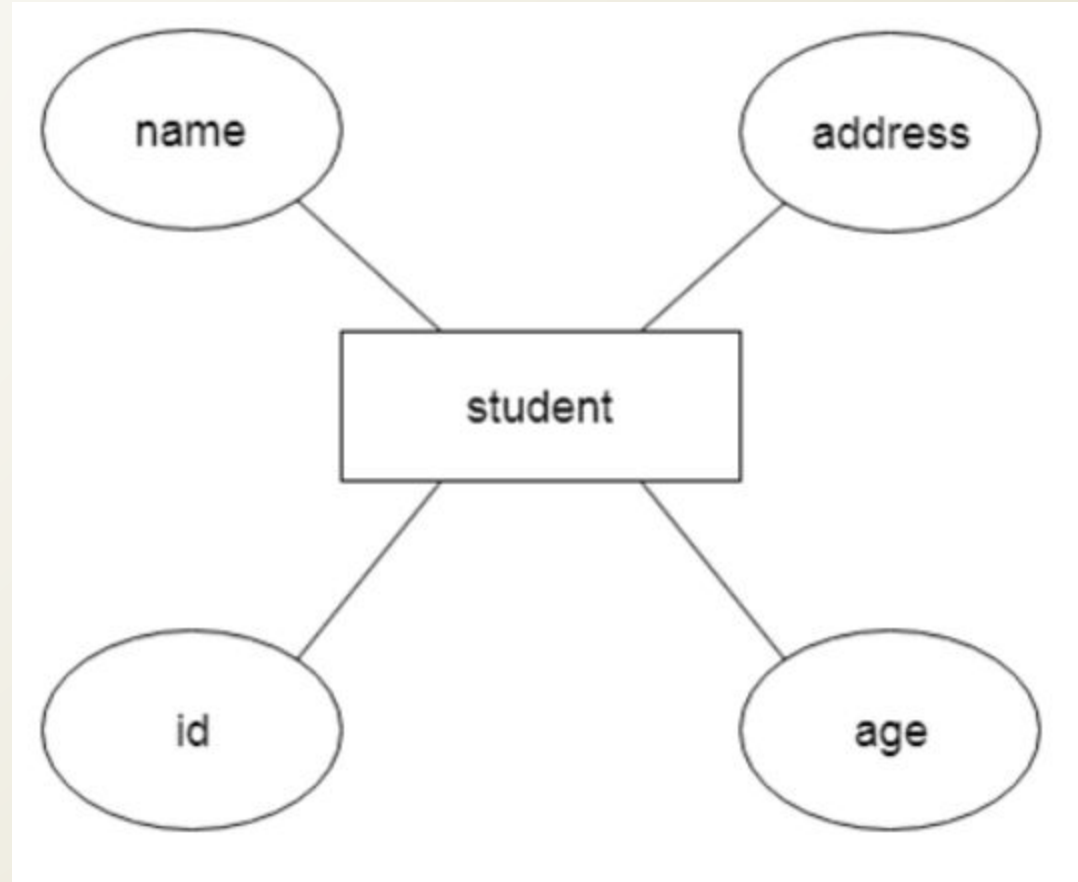
For example, Suppose we design a school database. In this database, the student will be an entity with attributes like address, name, id, age, etc. The address can be another entity with attributes like city, street name, pin code, etc and there will be a relationship between them.

ER Model:

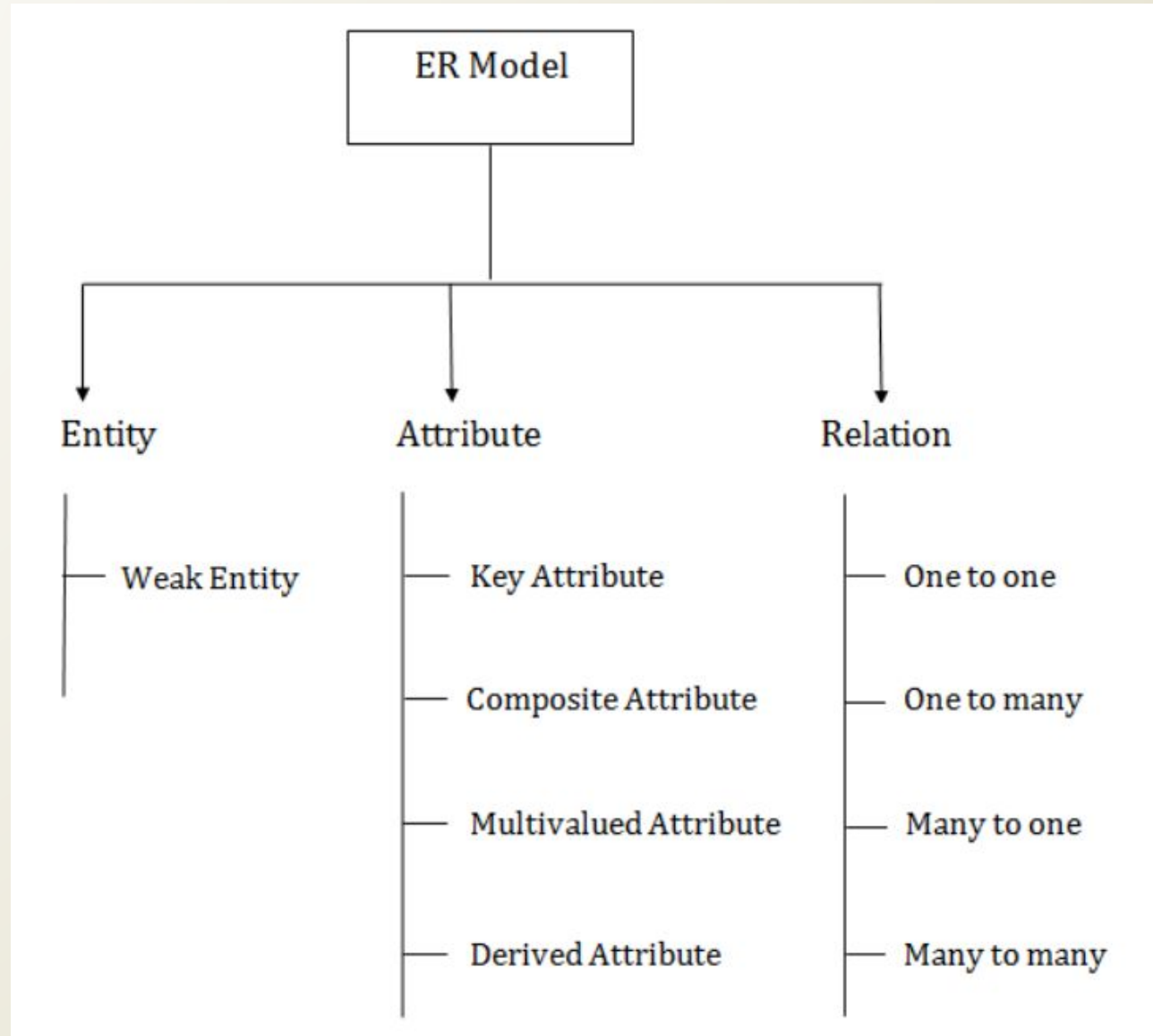
- ✓ ER model stands for an Entity-Relationship model. It is a high-level data model. This model is used to define the data elements and relationship for a specified system.
- ✓ It develops a conceptual design for the database. It also develops a very simple and easy to design view of data.
- ✓ In ER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

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.

ER Model:

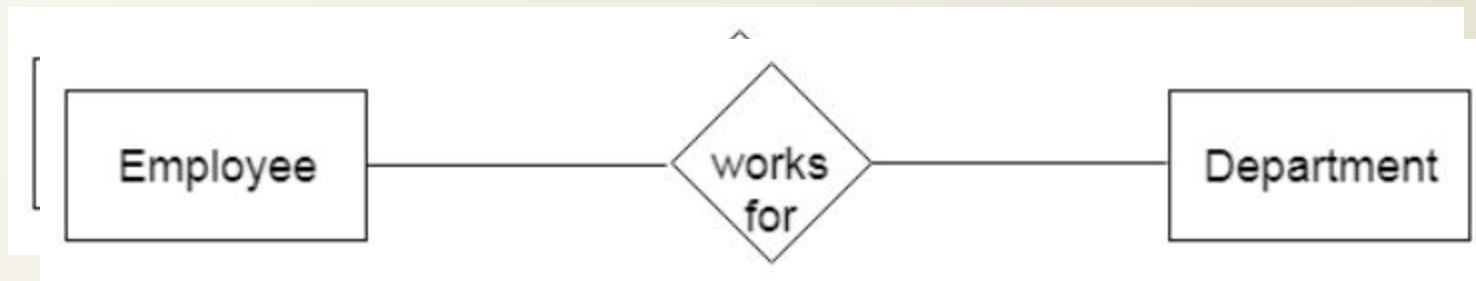


ER Model:



Entity:

- ✓ An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.
- ✓ Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.



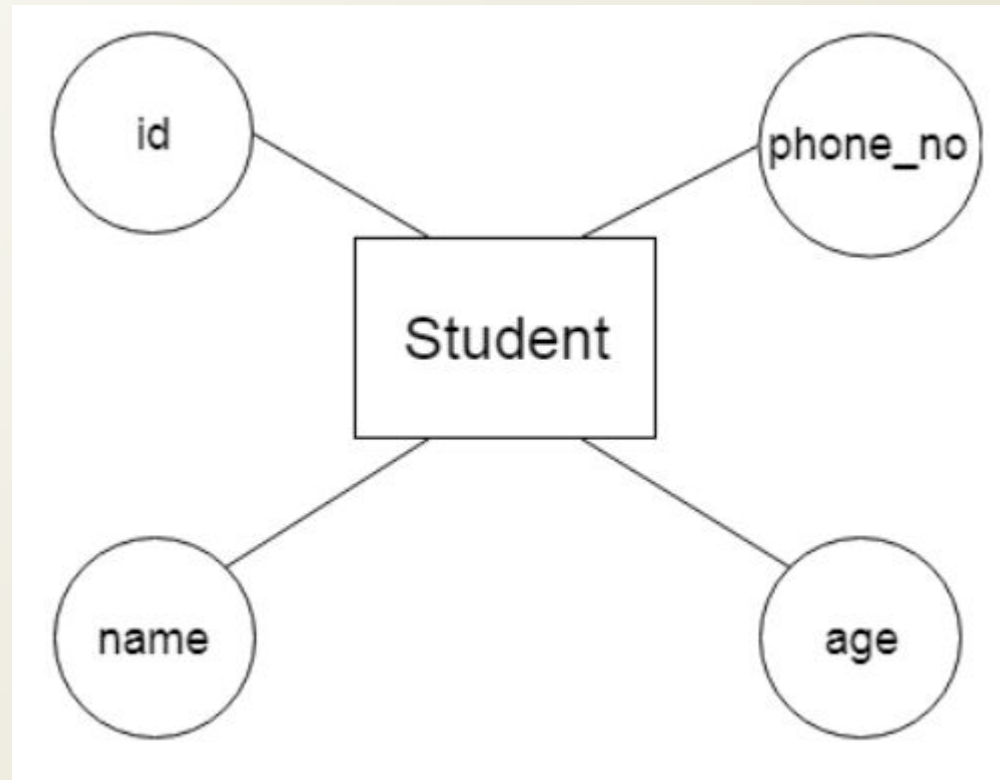
Weak Entity:

- ✓ An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.



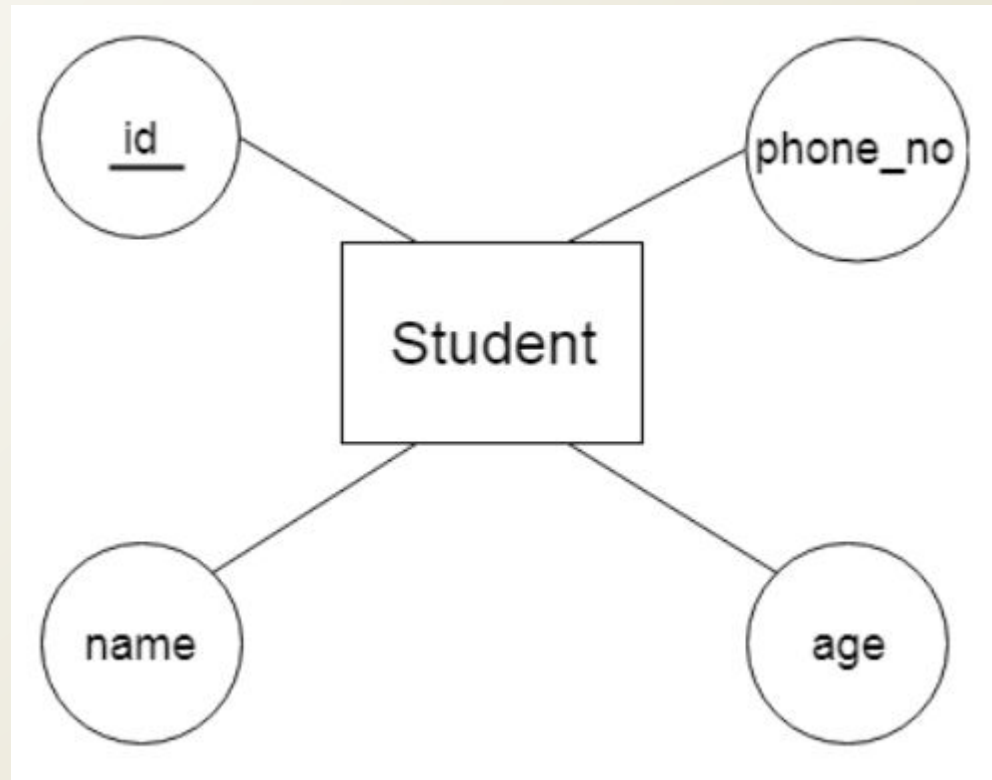
Attribute:

- ✓ The attribute is used to describe the property of an entity. Eclipse is used to represent an attribute.
- ✓ For example, id, age, contact number, name, etc. can be attributes of a student.



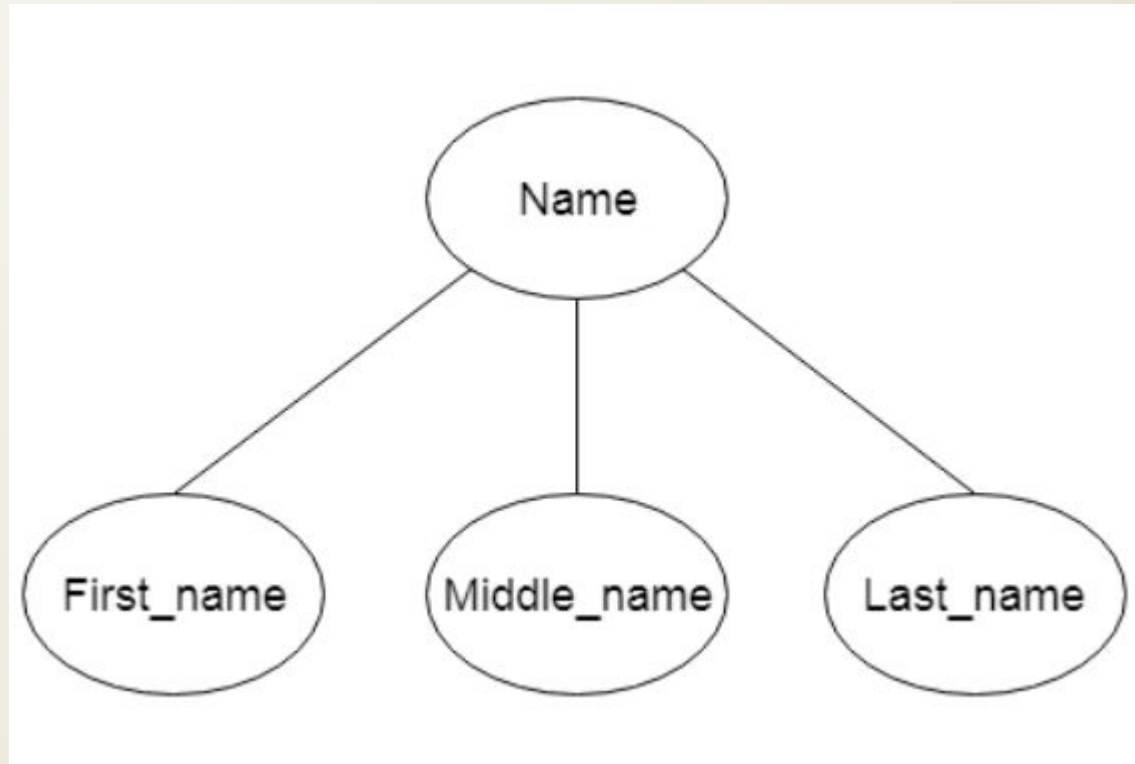
Key Attribute:

- ✓ The key attribute is used to represent the main characteristics of an entity. It represents a primary key. The key attribute is represented by an ellipse with the text underlined.



Composite Attribute:

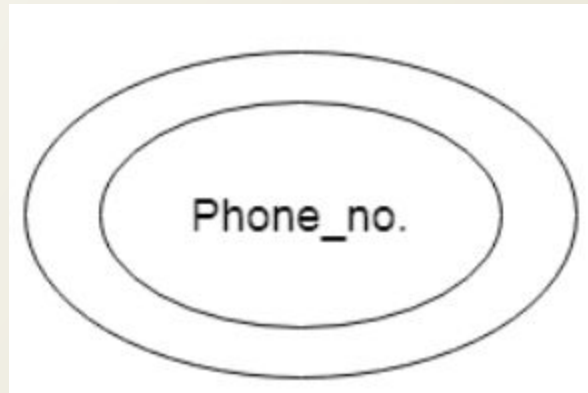
- ✓ An attribute that composed of many other attributes is known as a composite attribute. The composite attribute is represented by an ellipse, and those ellipses are connected with an ellipse.



Multivalued Attribute:

- ✓ An attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent multivalued attribute.

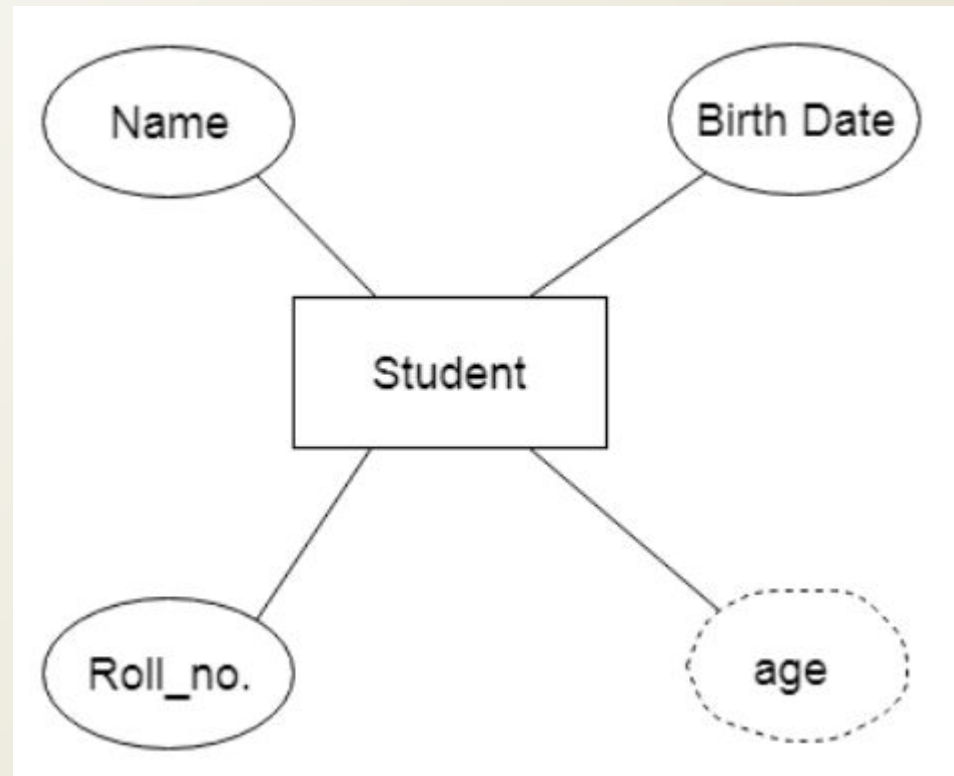
For example, a student can have more than one phone number.



Derived Attribute:

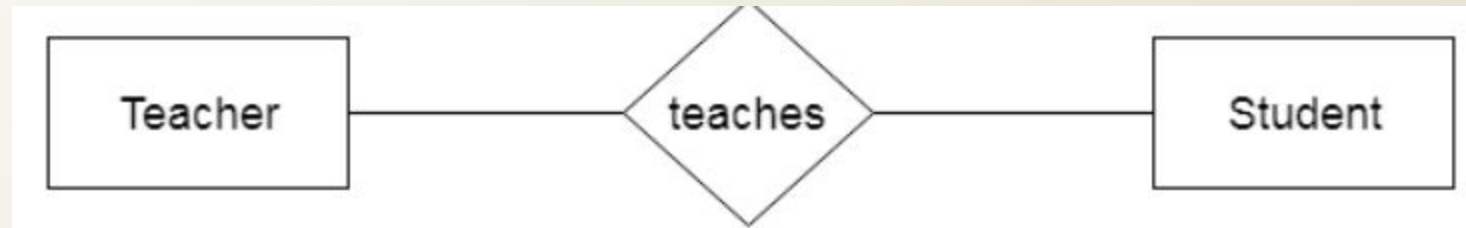
An attribute that can be derived from other attribute is known as a derived attribute. It can be represented by a dashed ellipse.

For example, A person's age changes over time and can be derived from another attribute like Date of birth.



Relationship:

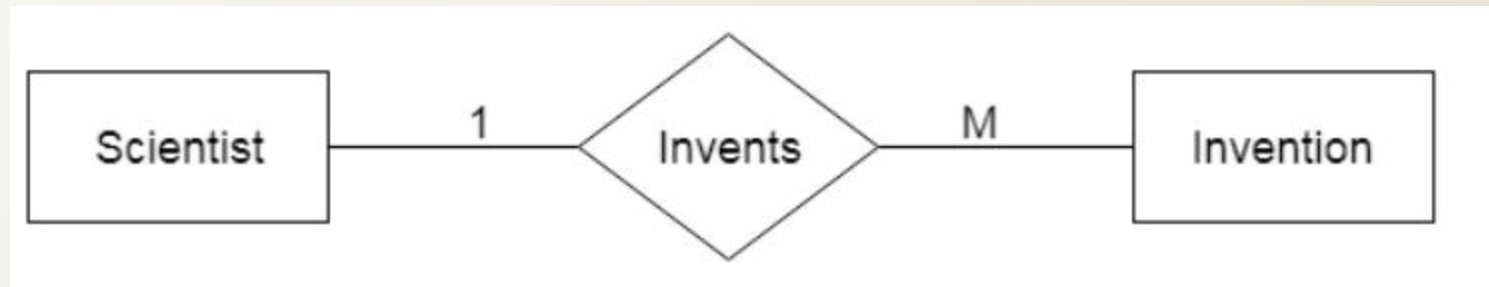
A relationship is used to describe the relation between entities. Diamond or rhombus is used to represent the relationship.



One-to-many relationship:

When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.

For example, Scientist can invent many inventions, but the invention is done by the only specific scientist.



One-to-One Relationship:

When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.

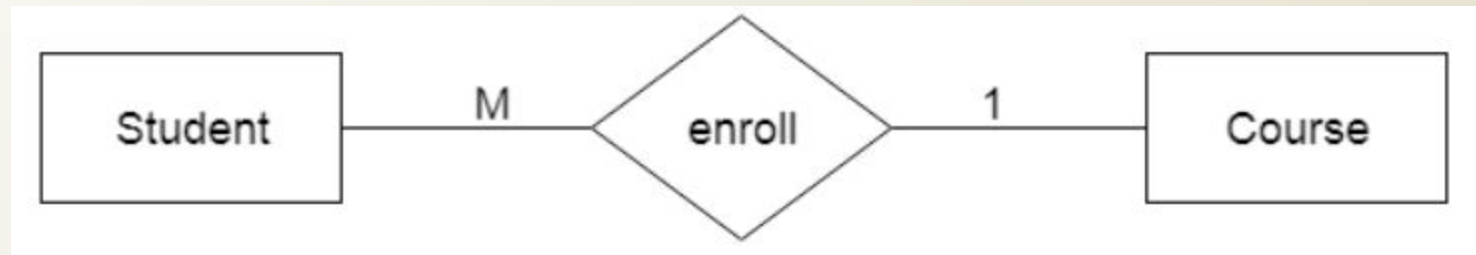
For example, A female can marry to one male, and a male can marry to one female.



Many-to-one relationship:

When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.

For example, Student enrolls for only one course, but a course can have many students.



Many-to-many relationship:

When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.

For example, Employee can assign by many projects and project can have many employees.



What is Normalization in DBMS (SQL)?:

- ✓ Normalization is the process of organizing the data in the database.
- ✓ Normalization is used to minimize the redundancy from a relation or set of relations. It is also used to eliminate undesirable characteristics like Insertion, Update, and Deletion Anomalies.
- ✓ Normalization divides the larger table into smaller and links them using relationships.
- ✓ The normal form is used to reduce redundancy from the database table.

Database Normal Forms:

Here is a list of Normal Forms in SQL:

- ✓ 1NF (First Normal Form)
- ✓ 2NF (Second Normal Form)
- ✓ 3NF (Third Normal Form)

Database Normalization With Examples

Database Normalization Example can be easily understood with the help of a case study. Assume, a video library maintains a database of movies rented out.

Without any normalization in database, all information is stored in one table as shown below. Let's understand Normalization database with normalization example with solution:

FULL NAMES	PHYSICAL ADDRESS	MOVIES RENTED	SALUTATION
Janet Jones	First Street Plot No 4	Pirates of the Caribbean, Clash of the Titans	Ms.
Robert Phil	3 rd Street 34	Forgetting Sarah Marshal, Daddy's Little Girls	Mr.
Robert Phil	5 th Avenue	Clash of the Titans	Mr.

Here you see Movies Rented column has multiple values. Now let's move into 1st Normal Forms:

1NF (First Normal Form) Rules:

- ✓ Each table cell should contain a single value.
- ✓ Each record needs to be unique.

The above table in 1NF-

FULL NAMES	PHYSICAL ADDRESS	MOVIES RENTED	SALUTATION
Janet Jones	First Street Plot No 4	Pirates of the Caribbean	Ms.
Janet Jones	First Street Plot No 4	Clash of the Titans	Ms.
Robert Phil	3 rd Street 34	Forgetting Sarah Marshal	Mr.
Robert Phil	3 rd Street 34	Daddy's Little Girls	Mr.
Robert Phil	5 th Avenue	Clash of the Titans	Mr.

What is a KEY in SQL?

Here is a list of Normal Forms in SQL:

- ✓ A KEY in SQL is a value used to identify records in a table uniquely.
- ✓ An SQL KEY is a single column or combination of multiple columns used to uniquely identify rows or tuples in the table.
- ✓ SQL Key is used to identify duplicate information, and it also helps establish a relationship between multiple tables in the database.
- ✓ Note: Columns in a table that are NOT used to identify a record uniquely are called non-key columns.

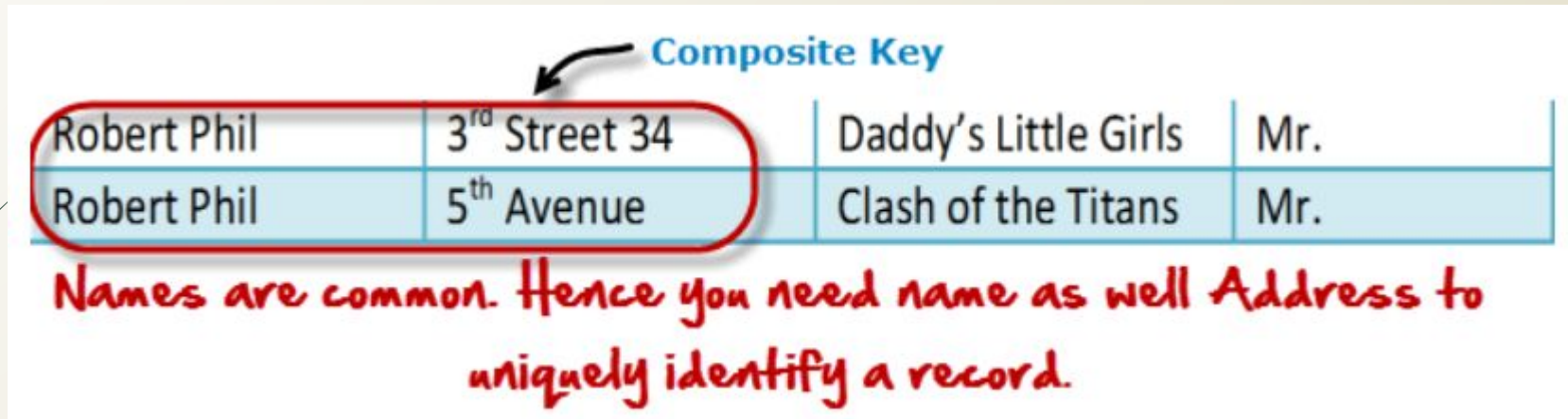
What is a Primary Key?

- ✓ A primary is a single column value used to identify a database record uniquely.
It has following attributes
- ✓ A primary key cannot be NULL
- ✓ A primary key value must be unique
- ✓ The primary key values should rarely be changed
- ✓ The primary key must be given a value when a new record is inserted.



What is Composite Key?

- ✓ A composite key is a primary key composed of multiple columns used to identify a record uniquely. In our database, we have two people with the same name Robert Phil, but they live in different places.



Robert Phil	3 rd Street 34	Daddy's Little Girls	Mr.
Robert Phil	5 th Avenue	Clash of the Titans	Mr.

Names are common. Hence you need name as well Address to uniquely identify a record.

Hence, we require both Full Name and Address to identify a record uniquely. That is a composite key. Let's move into second normal form 2NF

2NF (Second Normal Form) Rules:

- ✓ Rule 1- Be in 1NF
- ✓ Rule 2- Single Column Primary Key that does not functionally dependent on any subset of candidate key relation
- ✓ It is clear that we can't move forward to make our simple database in 2nd Normalization form unless we partition the table above.

MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION
1	Janet Jones	First Street Plot No 4	Ms.
2	Robert Phil	3 rd Street 34	Mr.
3	Robert Phil	5 th Avenue	Mr.

MEMBERSHIP ID	MOVIES RENTED
1	Pirates of the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans

2NF (Second Normal Form) Rules:

We have divided our 1NF table into two tables viz. Table 1 and Table2. Table 1 contains member information. Table 2 contains information on movies rented.

We have introduced a new column called Membership_id which is the primary key for table 1. Records can be uniquely identified in Table 1 using membership id

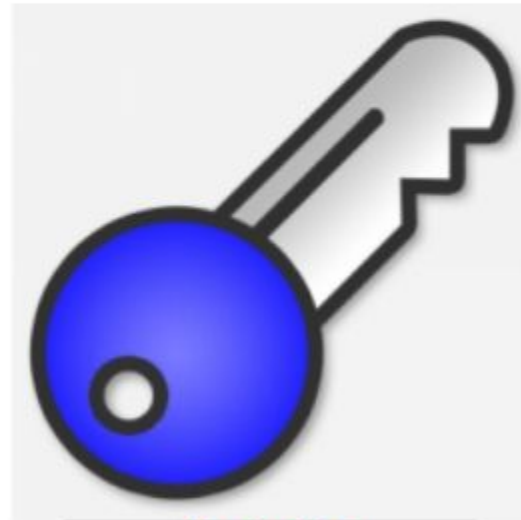
MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION
1	Janet Jones	First Street Plot No 4	Ms.
2	Robert Phil	3 rd Street 34	Mr.
3	Robert Phil	5 th Avenue	Mr.

MEMBERSHIP ID	MOVIES RENTED
1	Pirates of the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans

Database – Foreign Key?

In Table 2, Membership_ID is the Foreign Key.

MEMBERSHIP ID	MOVIES RENTED
1	Pirates of the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans



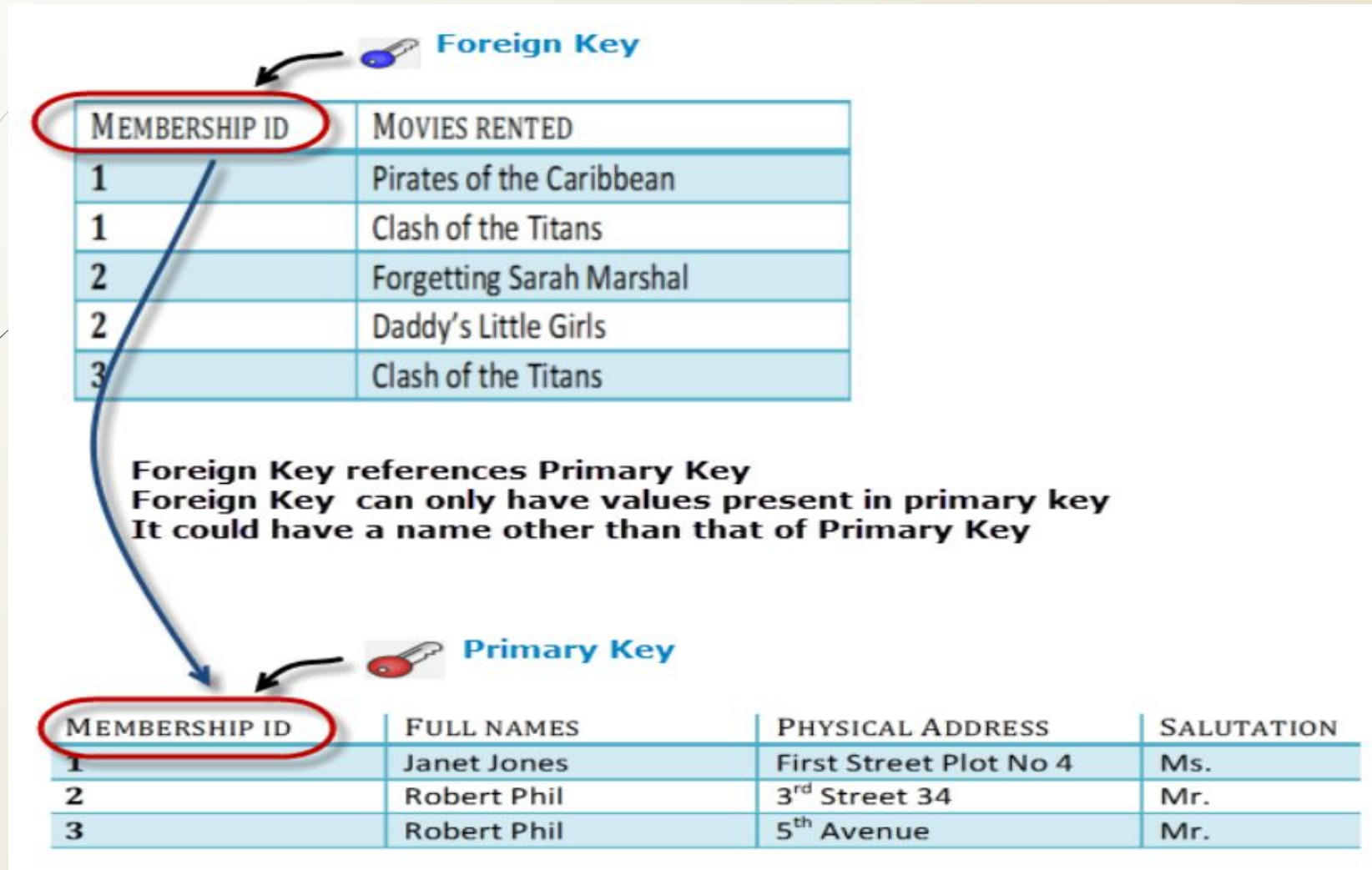
Foreign Key

Database – Foreign Key?

Foreign Key references the primary key of another Table! It helps connect your Tables

- ✓ A foreign key can have a different name from its primary key
- ✓ It ensures rows in one table have corresponding rows in another
- ✓ Unlike the Primary key, they do not have to be unique. Most often they aren't
- ✓ Foreign keys can be null even though primary keys can not

Database – Foreign Key?



Why do you need a foreign key?

Suppose, a novice inserts a record in Table B such as

Insert a record in Table 2 where Member ID = 101

MEMBERSHIP ID	MOVIES RENTED
101	Mission Impossible

But Membership ID 101 is not present in Table 1

MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION
1	Janet Jones	First Street Plot No 4	Ms.
2	Robert Phil	3 rd Street 34	Mr.
3	Robert Phil	5 th Avenue	Mr.

Database will throw an **ERROR**. This helps in referential integrity

Why do you need a foreign key?

- ✓ You will only be able to insert values into your foreign key that exist in the unique key in the parent table. This helps in referential integrity.
- ✓ The above problem can be overcome by declaring membership id from Table2 as foreign key of membership id from Table1
- ✓ Now, if somebody tries to insert a value in the membership id field that does not exist in the parent table, an error will be shown!

What are transitive functional dependencies?

- ✓ A transitive functional dependency is when changing a non-key column, might cause any of the other non-key columns to change
- ✓ Consider the table 1. Changing the non-key column Full Name may change Salutation.

MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION
1	Janet Jones	First Street Plot No 4	Ms.
2	Robert Phil	3 rd Street 34	Mr.
3	Robert Phil	5 th Avenue	Mr.

Change in Name → *May Change Salutation*

3NF (Third Normal Form) Rules

- ✓ Rule 1- Be in 2NF
- ✓ Rule 2- Has no transitive functional dependencies
- ✓ To move our 2NF table

MEMBERSHIP ID	FULL NAMES	PHYSICAL ADDRESS	SALUTATION ID
1	Janet Jones	First Street Plot No 4	2
2	Robert Phil	3 rd Street 34	1
3	Robert Phil	5 th Avenue	1

MEMBERSHIP ID	MOVIES RENTED
1	Pirates of the Caribbean
1	Clash of the Titans
2	Forgetting Sarah Marshal
2	Daddy's Little Girls
3	Clash of the Titans

SALUTATION ID	SALUTATION
1	Mr.
2	Ms.
3	Mrs.
4	Dr.

3NF (Third Normal Form) Rules

We have again divided our tables and created a new table which stores Salutations.

- ✓ There are no transitive functional dependencies, and hence our table is in 3NF
- ✓ In Table 3 Salutation ID is primary key, and in Table 1 Salutation ID is foreign to primary key in Table 3