VIPS
Technical Campus
योग: कर्मसु कौशलम्
IN PURSUIT OF PERFECTION

SCHOOL OF ENGINEERING AND TECHNOLOGY

VIPS

योगः कर्मसु कौशलम्

IN PURSUIT OF PERFECTION

Database Management Systems

Database

- What is a database?
  - A collection of files that store related data
- We will mainly focus on **relational** databases (i.e., data is stored in tables)

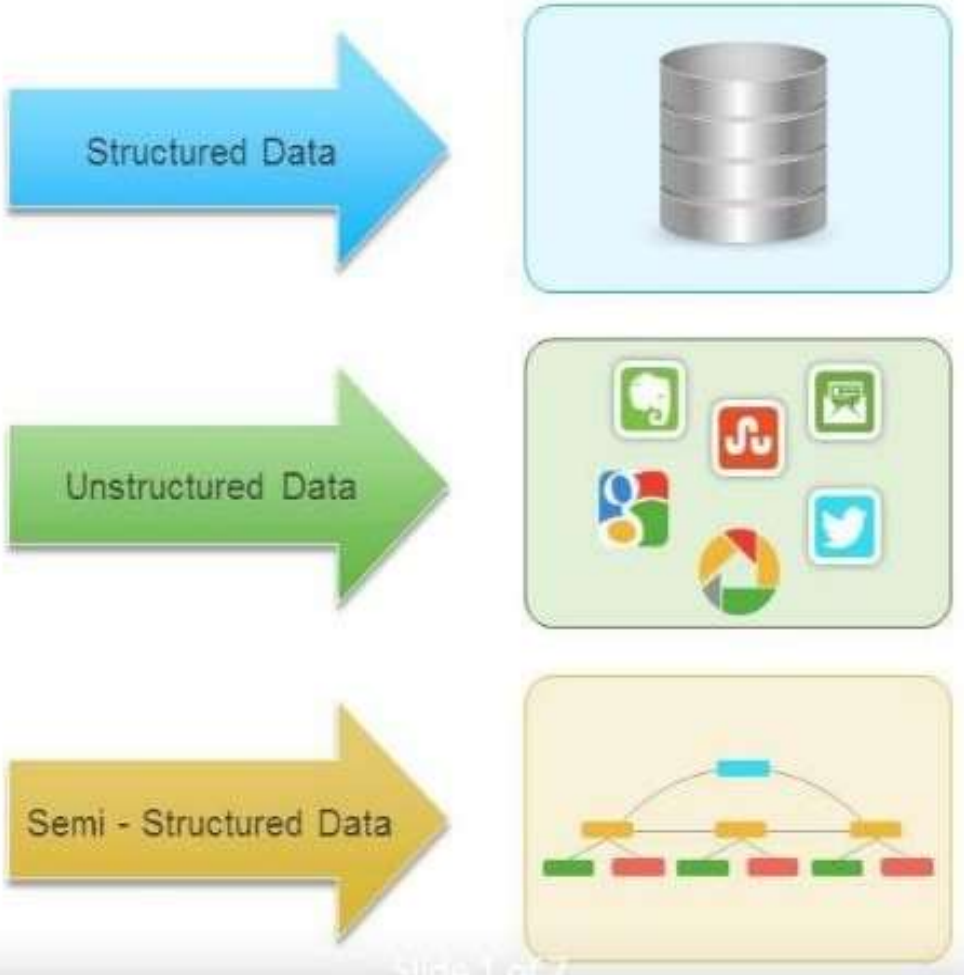
Name	Gender	GPA
Mike	Male	4.0
Bob	Male	3.6
Alice	Female	3.8

Text Data

Photos & Videos

Structured, Semi-structured and Unstructured data

FEATURES	STRUCTURED	SEMI STRUCTURED	UNSTRUCTURED
Format Type	Relational Database	HTML, XML, JSON	Binary, Character
Version Management	Rows, columns, tuples	Not as common – graph is possible	Whole data
Implementation	SQL	Anonymous nodes	-
Robustness	Robust	Limited robustness	-
Storage Requirement	Less	Significant	Large
Applications	DBMS, RDF, ERP system, Data Warehouse, Apache Parquet, Financial Data, Relational Table	Server Logs, Sensor Output	No SQL, Video, Audio, Social Media, Online Forums, MRI, Ultrasound



Structured vs Semi-Structured data vs Unstructured data

Structured data

- Represented in a strict format
- It has been organized into a formatted repository that is typically a database.
- It concerns all data which can be stored in database SQL in a table with rows and columns
- . *Example:* Relational data

Semi-Structured data

- Information that does not reside in a relational database but that have some organizational properties that make it easier to analyze
- With some process, you can store them in the relation database
- but Semi-structured exist to ease space.*Example*: XML data

Unstructured data

- Data which is not organized in a predefined manner or does not have a predefined data model
- thus it is not a good fit for a mainstream relational database.
- there are alternative platforms for storing and managing, used by organizations in a variety of business intelligence and analytics applications.
- ***Example*: Word, PDF, Text, Media logs.**

Data, Database & DBMS

Data

1. Known facts that can be recorded and have implicit meaning.
2. **Data** is nothing but facts and statistics stored or free flowing over a network, generally it's raw and unprocessed.
3. Data becomes **information** when it is processed, turning it into something meaningful.

Database

✓The collection of data.

✓A **Database** is a collection of **related data** organised in a way that data can be easily accessed, managed and updated.

Database-management system (DBMS)

✓ is a collection of interrelated data and a set of programs to access those data.

✓ General purpose software system that facilitates process of defining, constructing, manipulating, and sharing database

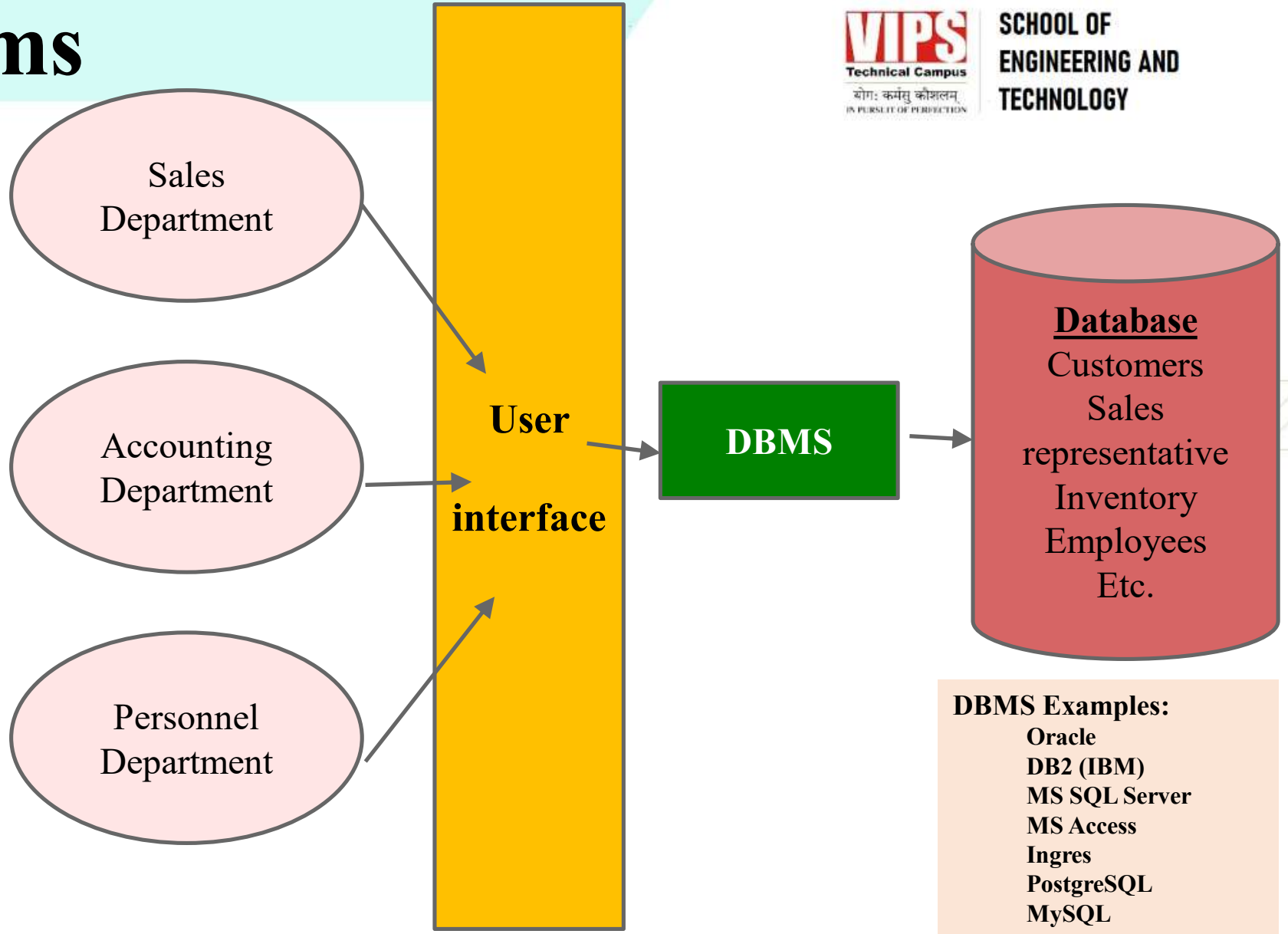
Database Management Systems (DBMSs)

- A piece of software designed to store and manage databases.
- A **DBMS** is a software that allows **creation, definition and manipulation of database, allowing users to store, process and analyse data easily.**
- DBMS also provides protection and **security** to the databases.
- It also maintains **data consistency** in case of multiple users.
- Database systems are designed to manage large bodies of information.
- Management of data involves both storage of information and mechanisms for manipulation of information.
- The database system must ensure the safety of the information stored.
- If data are to be shared among several users, the system must avoid possible anomalous results.
- Here are some examples of popular DBMS used these days:
  - **MySql, Oracle, SQL Server, IBM DB2, Amazon SimpleDB (cloud based) etc.**

Database Systems

- A database system consists of
  - Data (the database)
  - Software
  - Hardware
  - Users
- We focus mainly on the software
- Database systems allow users to
  - Store
  - Update
  - Retrieve
  - Organise
  - Protect

  their data.



DBMS Examples:
- Oracle
- DB2 (IBM)
- MS SQL Server
- MS Access
- Ingres
- PostgreSQL
- MySQL

Databases Applications

- Web indexes
- Library catalogues
- Medical records
- Bank accounts
- Stock control
- Personnel systems
- Product catalogues
- Telephone directories

- Train timetables
- Airline bookings
- Credit card details
- Student records
- Customer histories
- Stock market prices
- Discussion boards
- and so on…

The primary goal of a DBMS is to provide a way to store and retrieve database information that is both *convenient* and *efficient*.

Database Management System (DBMS)

- **DBMS contains information about a particular enterprise**
  - Collection of interrelated data
  - Set of programs to access the data
  - An environment that is both *convenient* and *efficient* to use
- **Database Applications:**
  - Banking: all transactions
  - Airlines: reservations, schedules
  - Universities: registration, grades
  - Sales: customers, products, purchases
  - Online retailers: order tracking, customized recommendations
  - Manufacturing: production, inventory, orders, supply chain
  - Human resources: employee records, salaries, tax deductions
- Databases touch all aspects of our lives

Purpose of Database Systems (Cont.)

Purpose of Database Systems

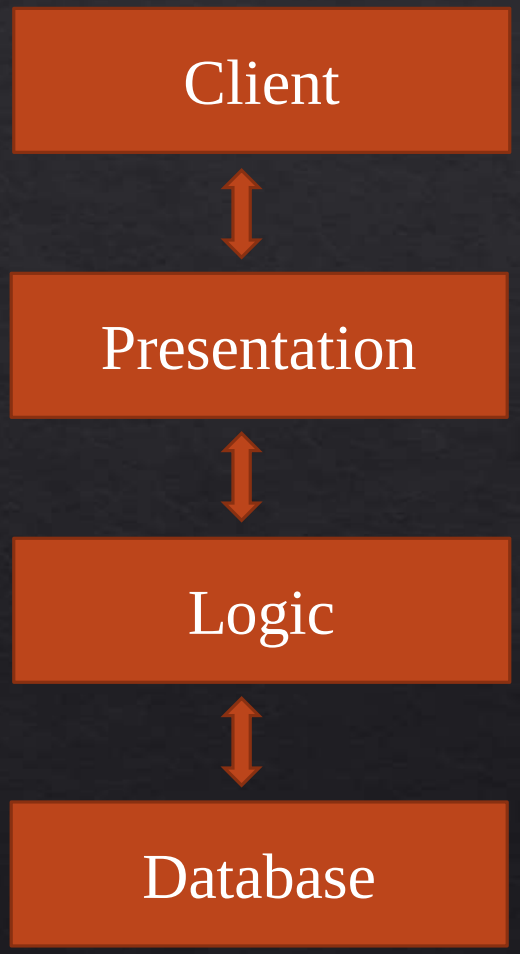
In the early days, database applications were built directly on top of file systems

Drawbacks of using file systems to store data:

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

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

Let’s Understand The Role of DBMS



Access the web app

Example: IE, Firefox, iPhone, Chrome, Android

Renders information, generates webpages

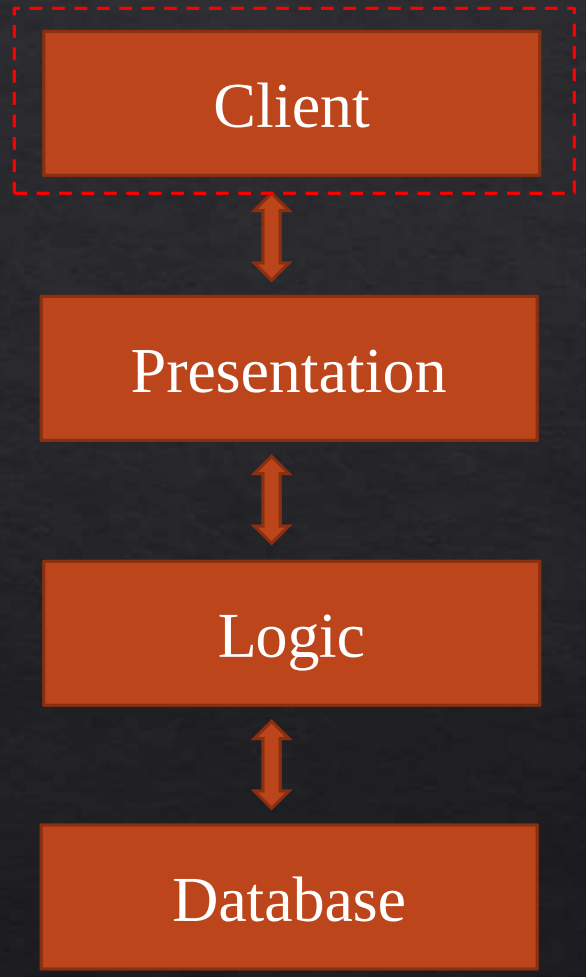
Example: HTML, DHTML, XML, Javascript

Manage data access, performs calculations, makes logical decisions

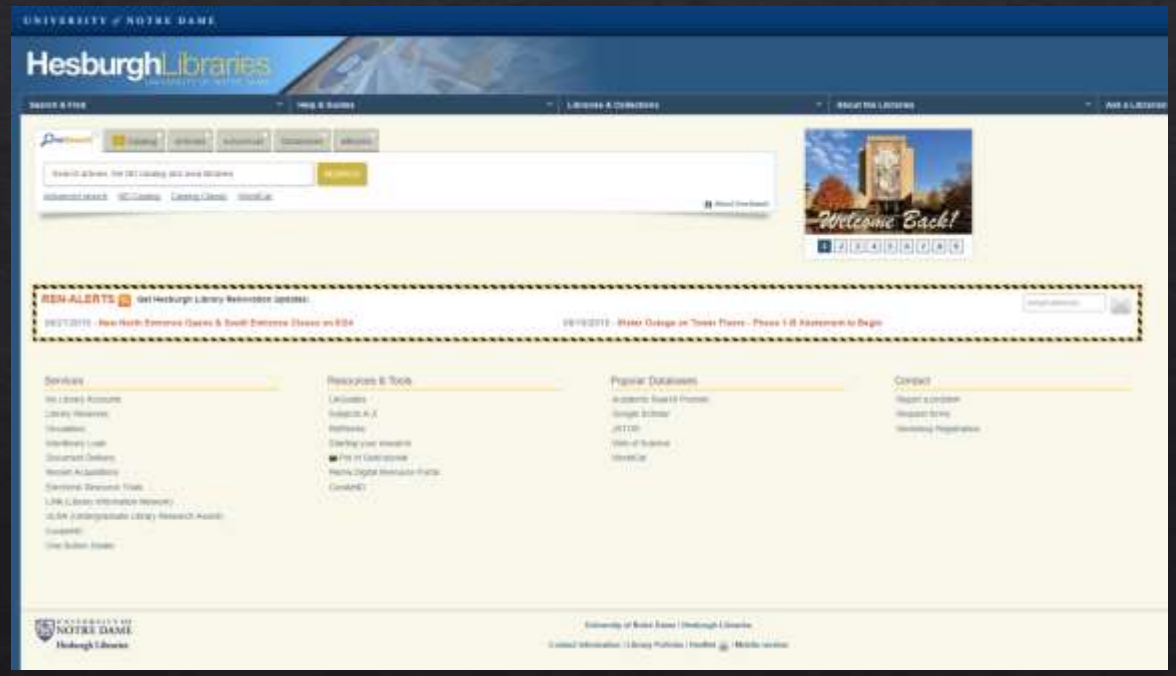
Example: ASP, PHP, Perl

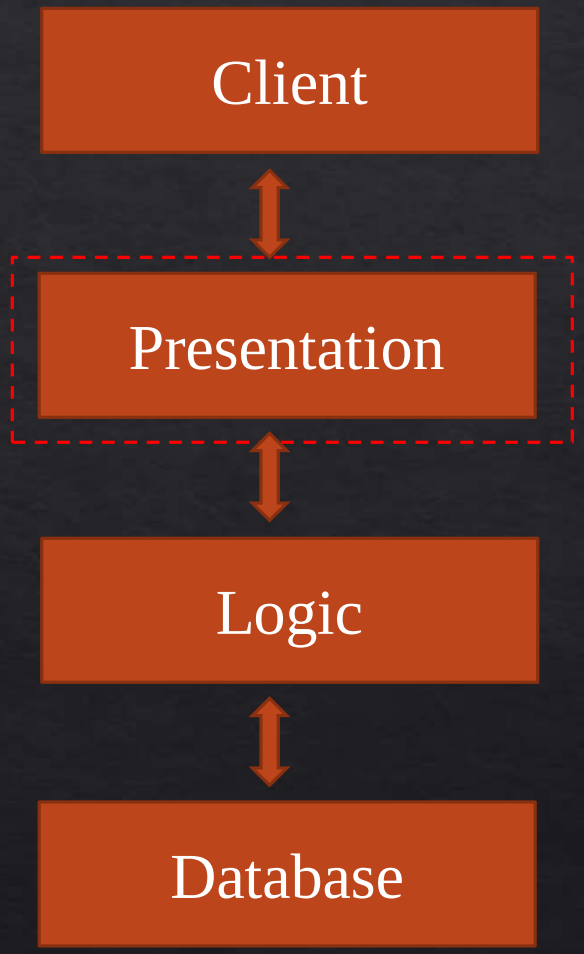
Stores and manages data

Example: MySQL, Oracle

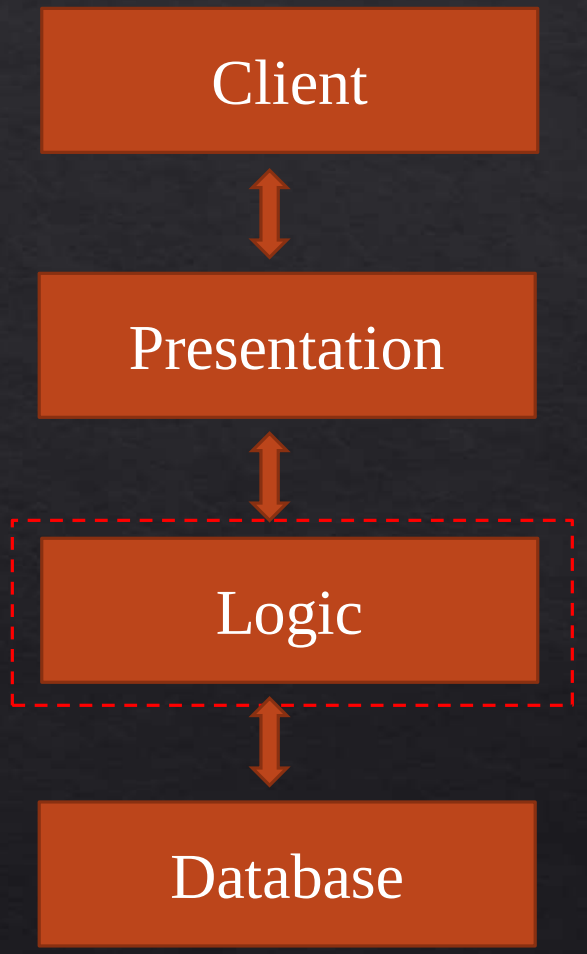


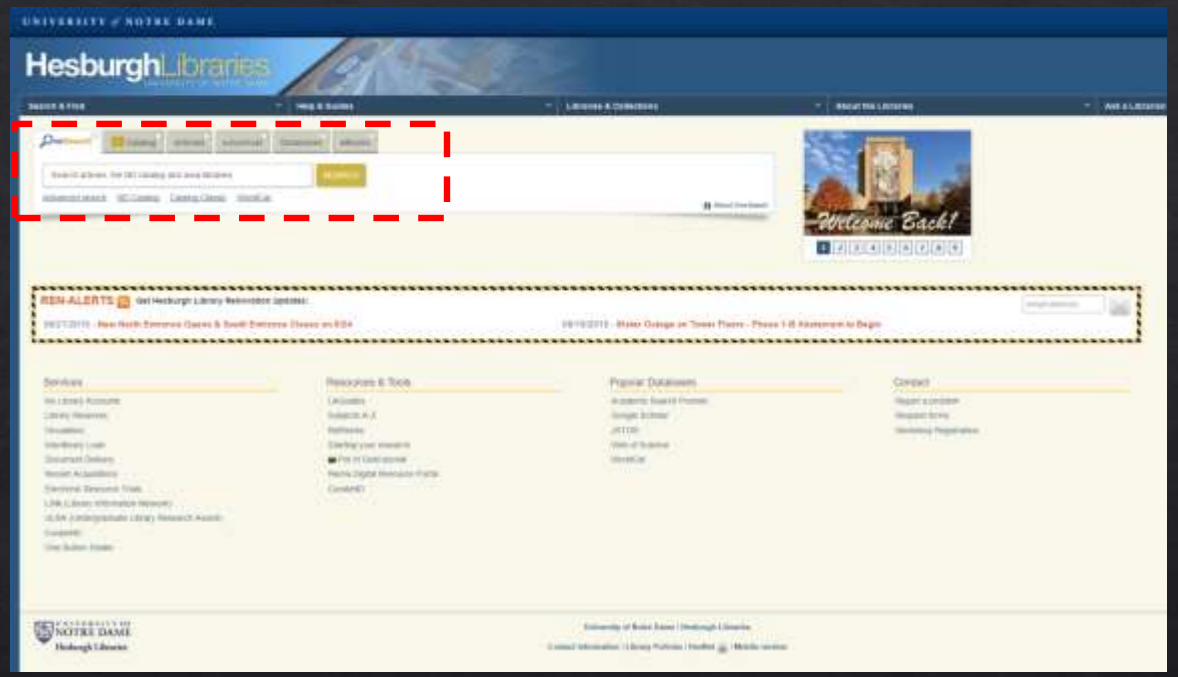
Application is a website accessible via a web browser:



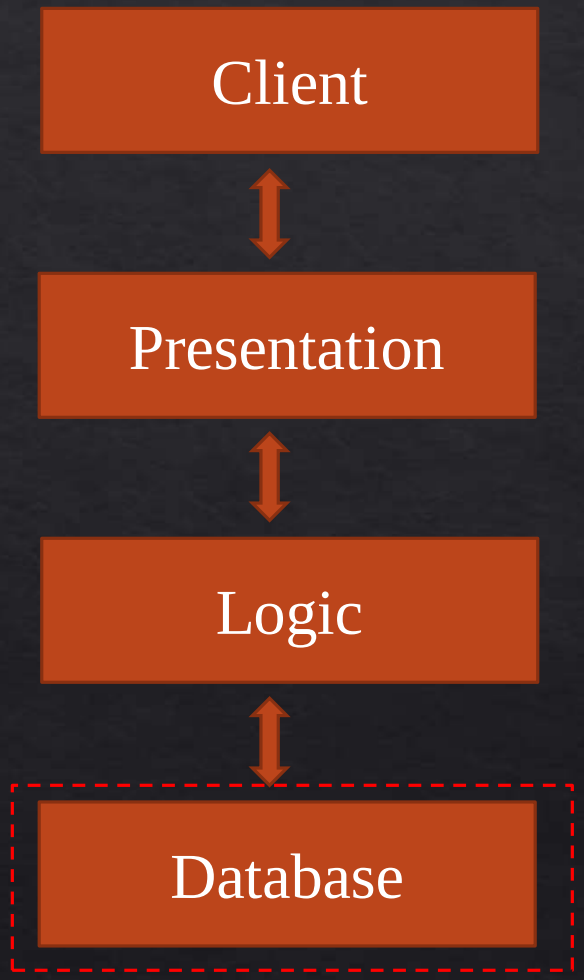


Use HTML and Javascript to create Web site

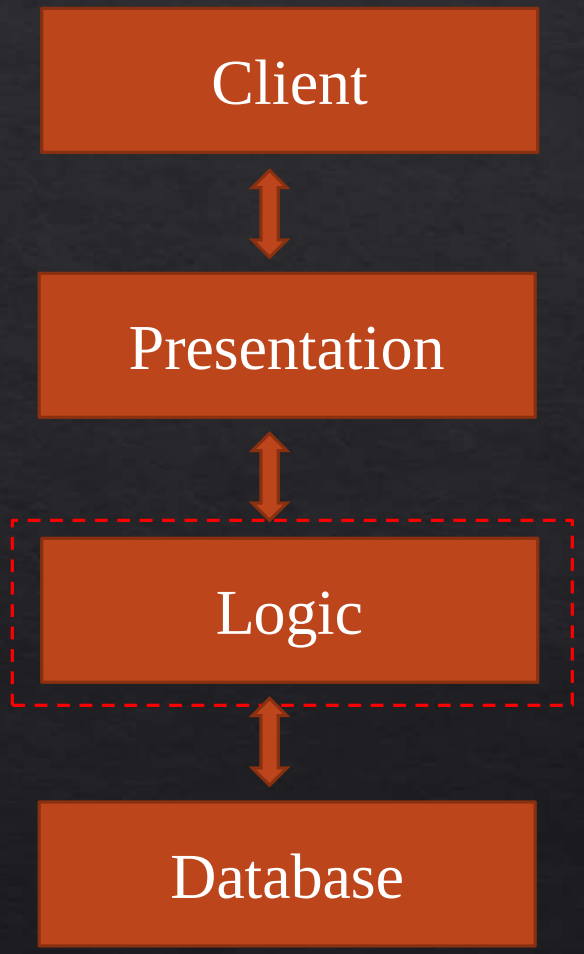


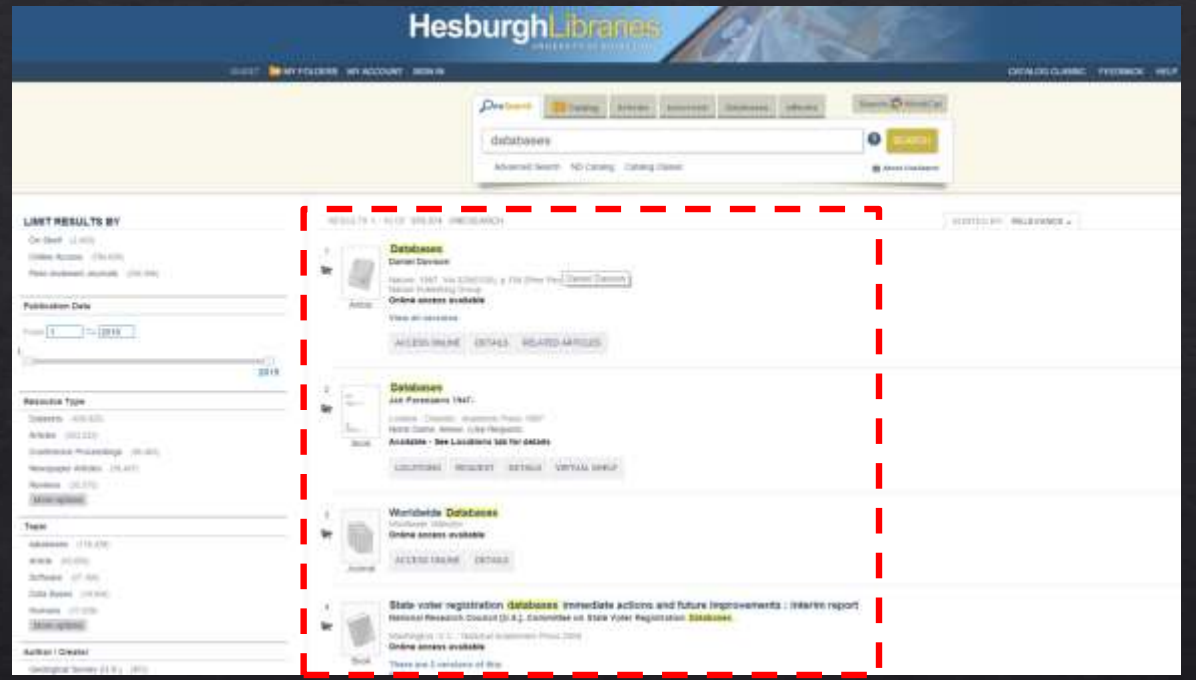


PHP, et al., takes a string input and makes a SQL query

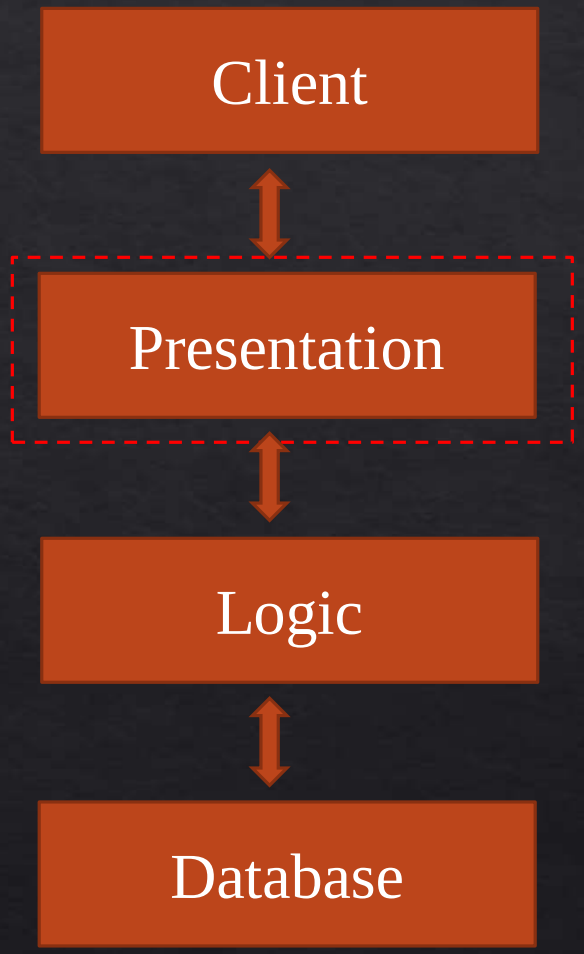


Mysql processes the query and returns results

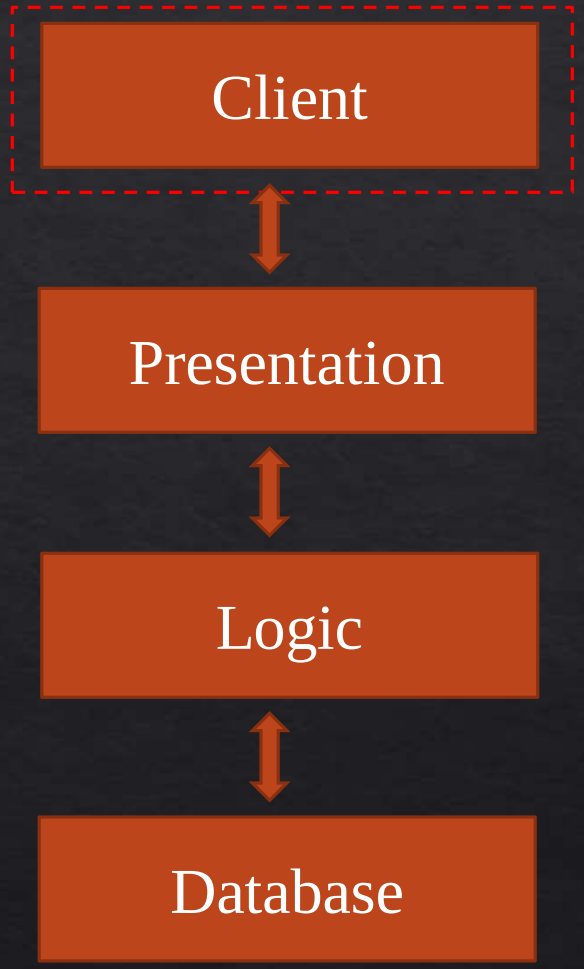




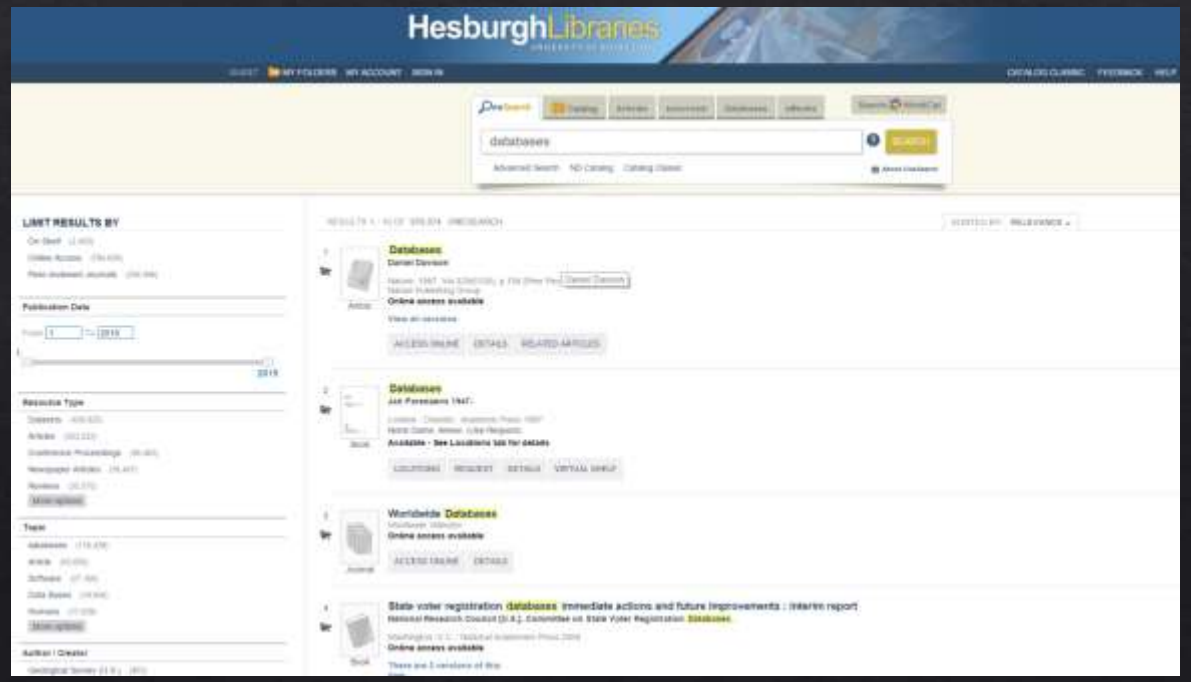
PHP, et al., processes the results, i.e., turns database objects into book records

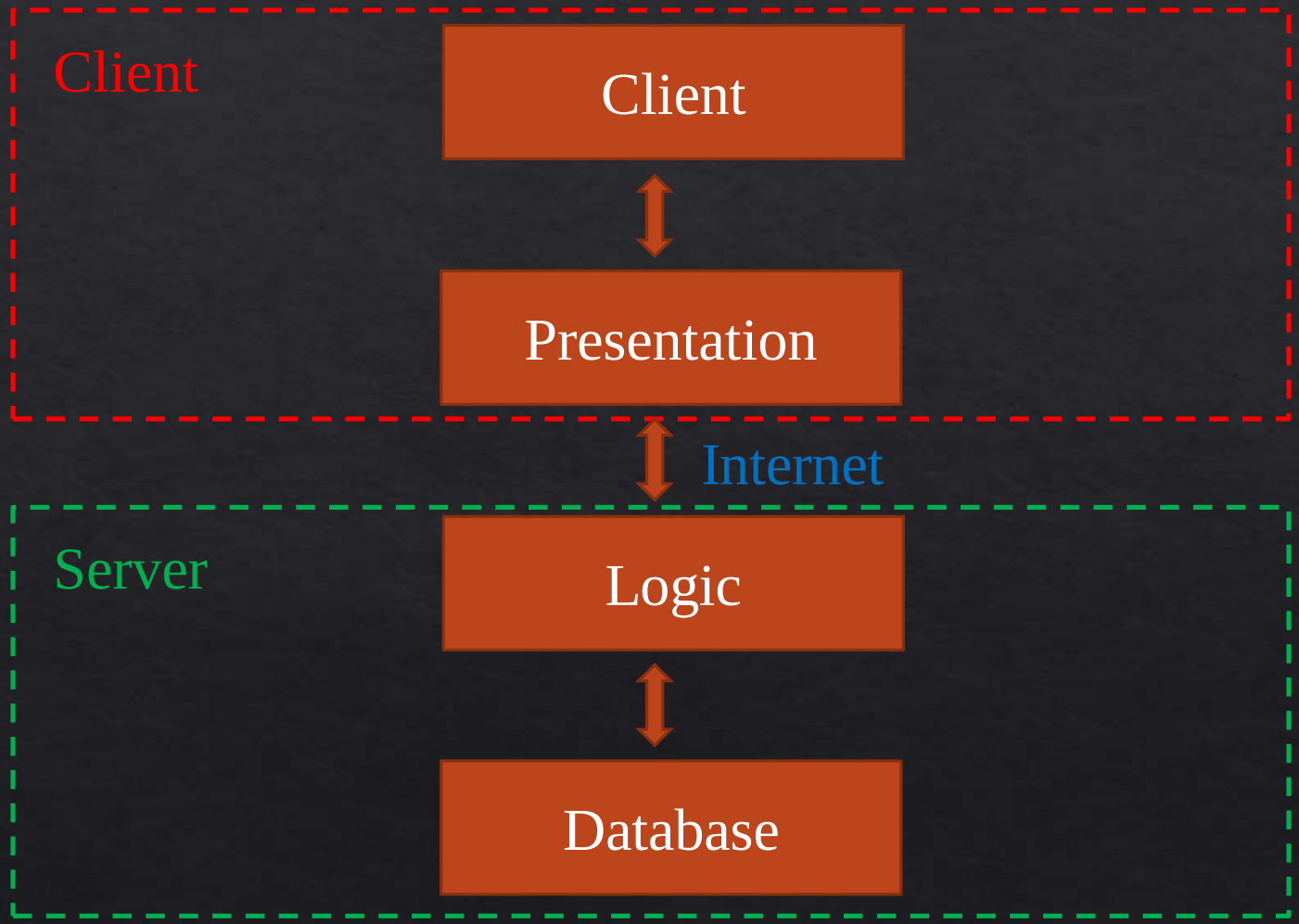


HTML and Javascript render the results

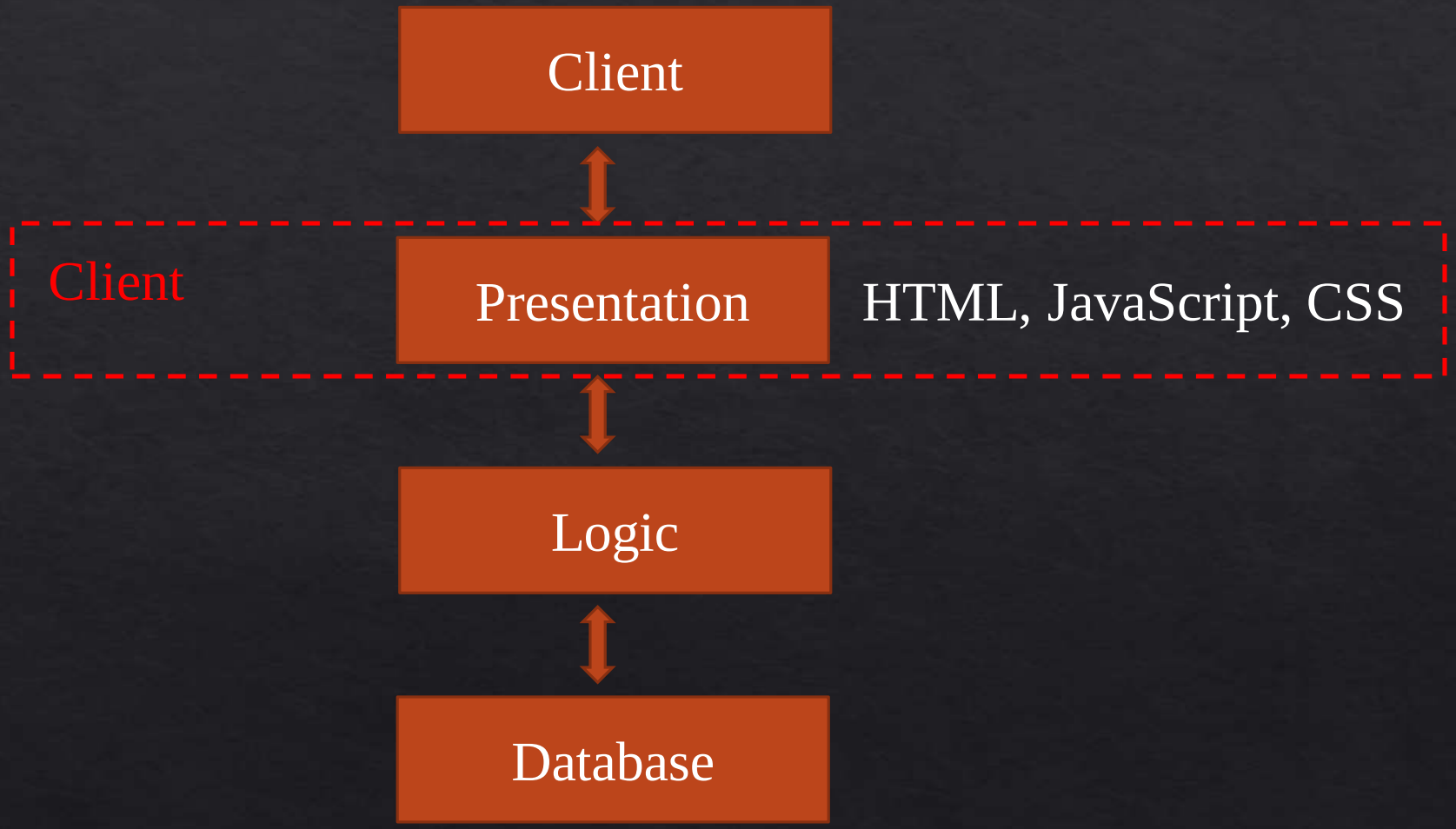


Client displays the results

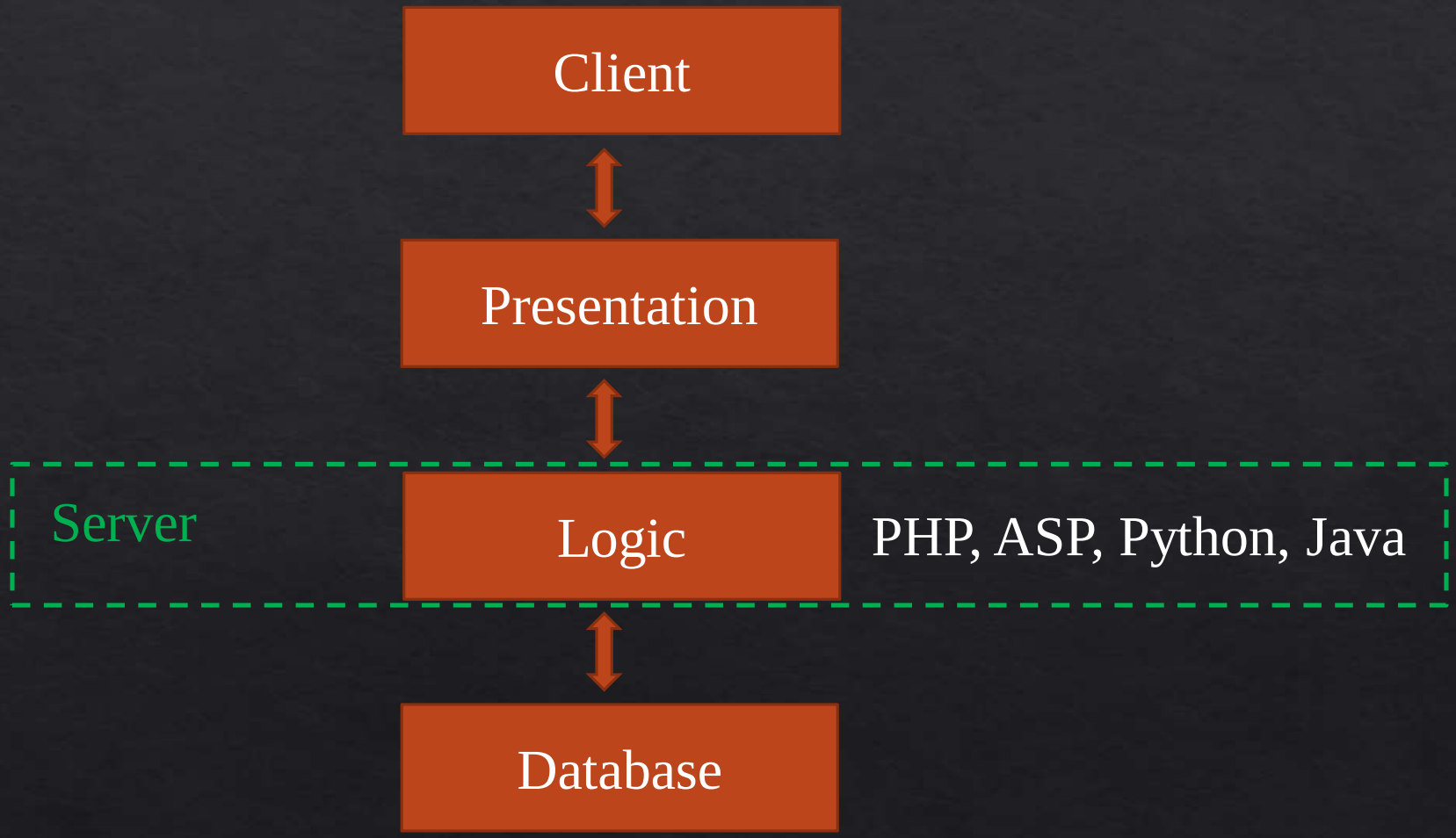



Client
Client
Presentation
Internet
Server
Logic
Database

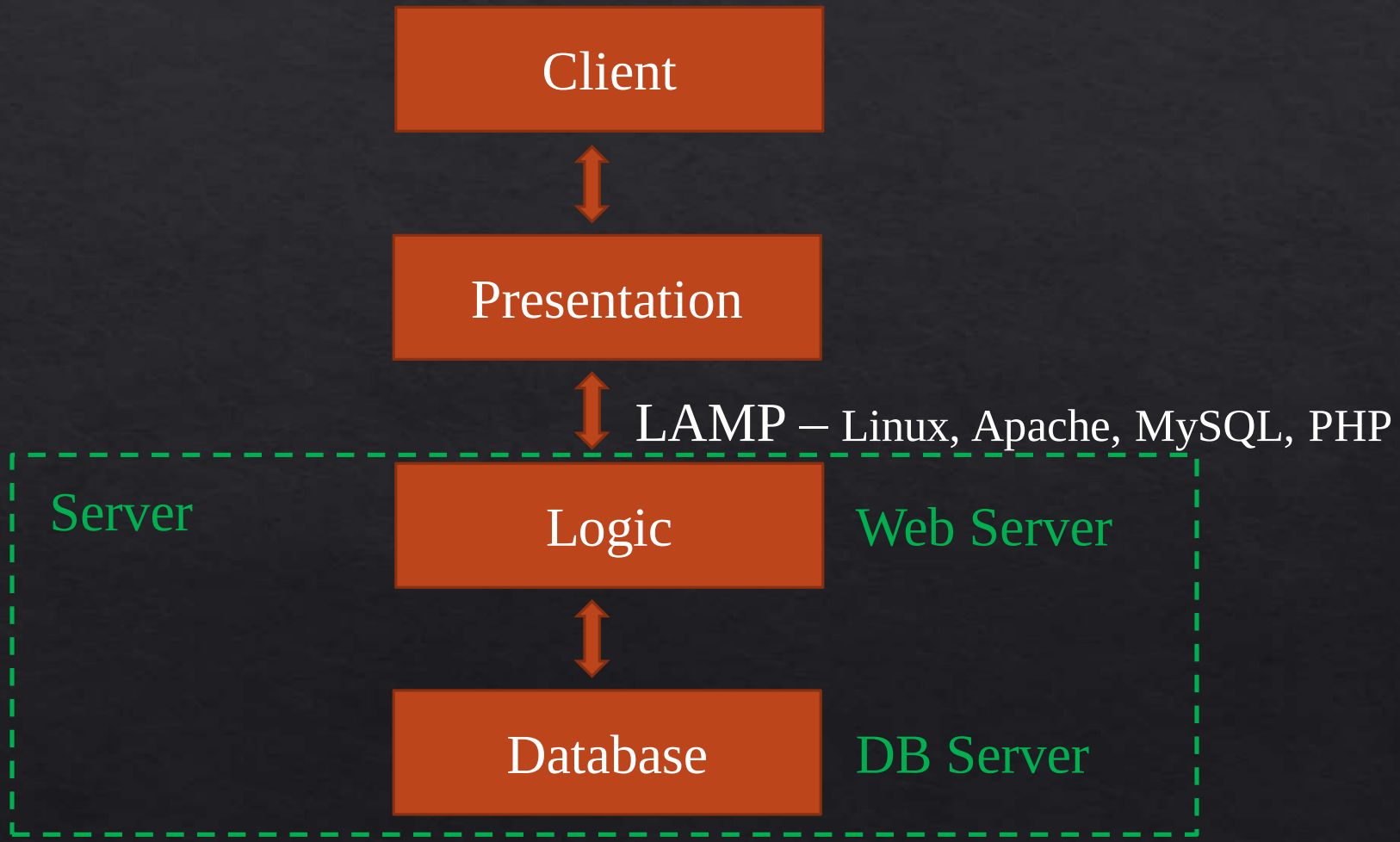
How to implement a Web application



How to implement a Web application



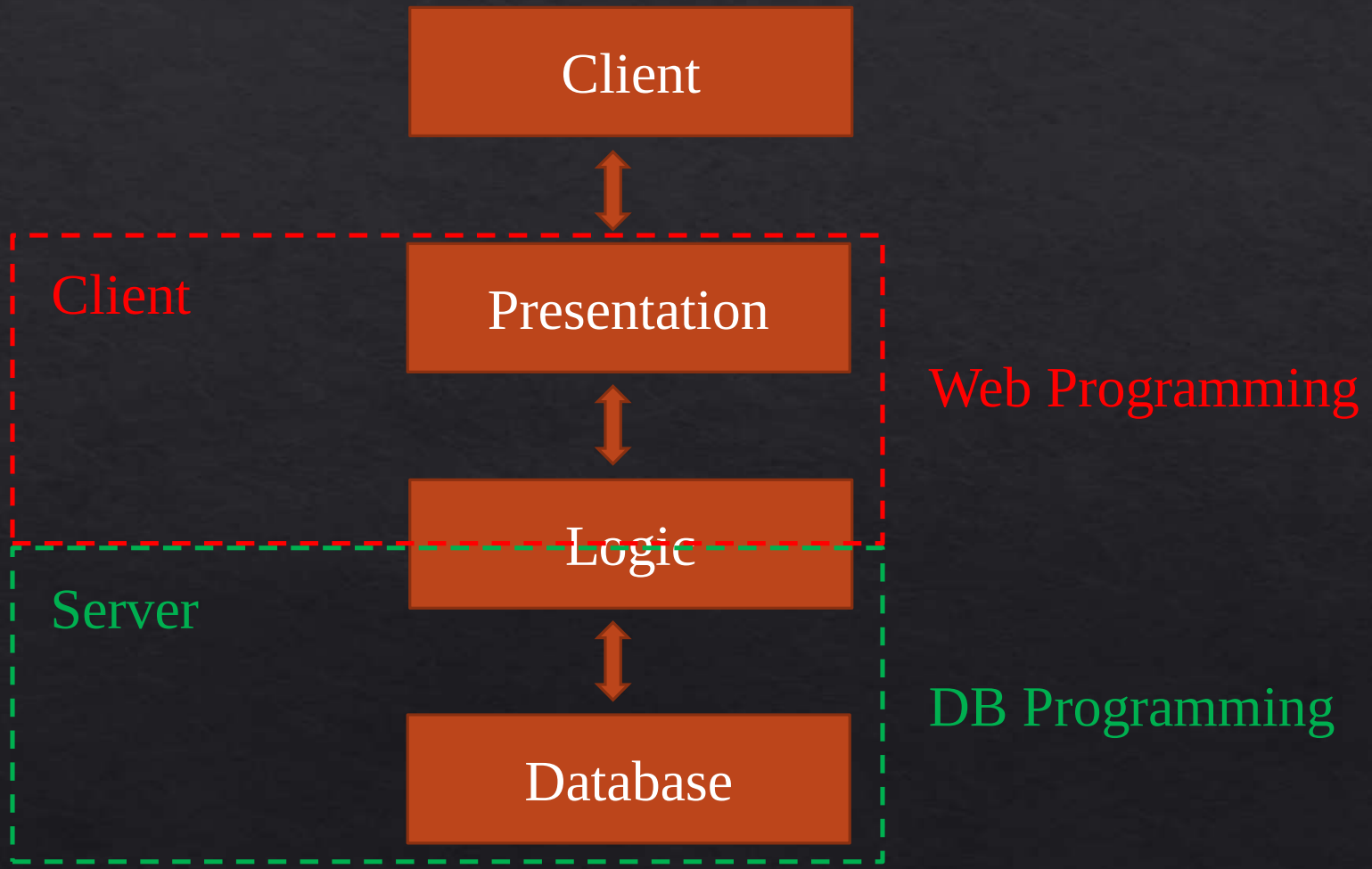
How to implement a Web application



Common tools for building web apps

- Framework:
  - LAMP
  - Client side:
    - Examples: CSS, Flash, HTML, DHTML, Javascipt
  - Server side:
    - Examples: ASP, Java, Perl, PHP, Python, Ruby
  - Web server:
    - Examples: Apache, Tomcat
  - Database:
    - Examples: MySQL, Oracle, Postgres

We will help you with project tutorials

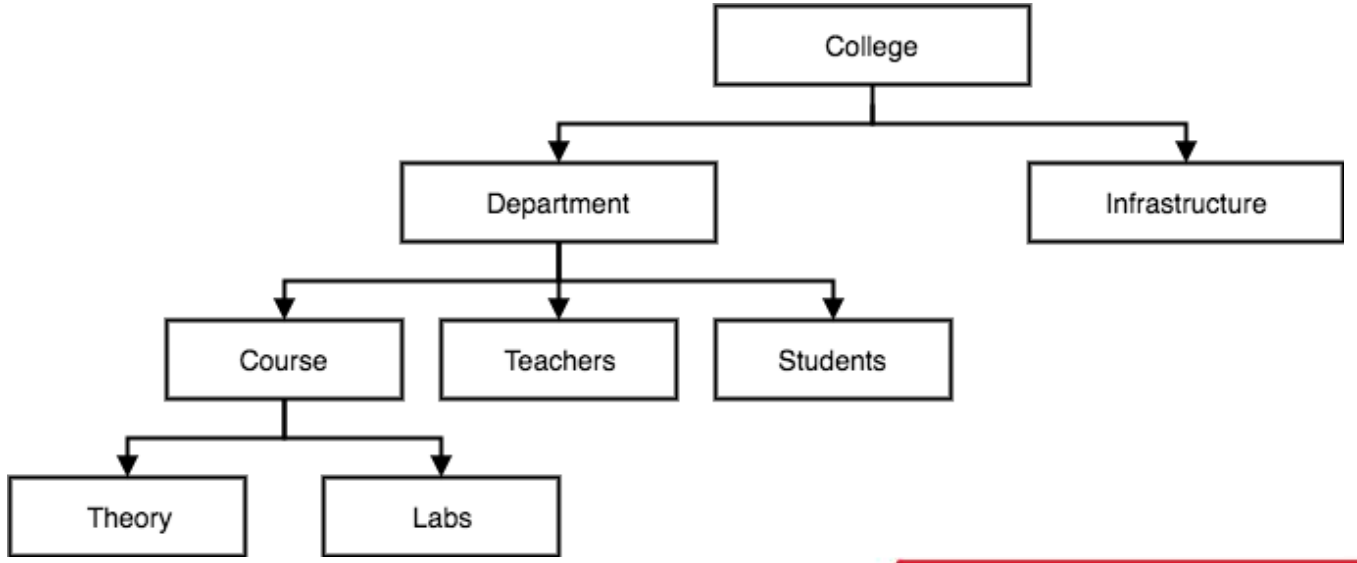


Data Models

Data Model is the modeling of the data description, data semantics, and consistency constraints of the data. It provides the conceptual tools for describing the design of a database at each level of data abstraction.

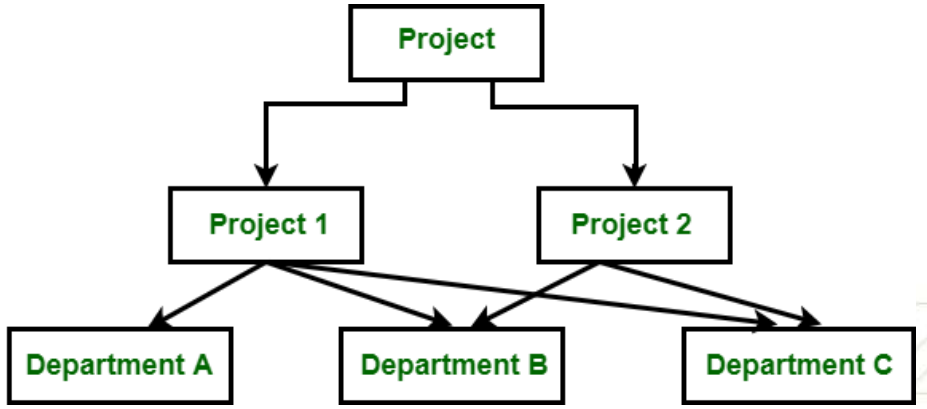
Hierarchical Database Model

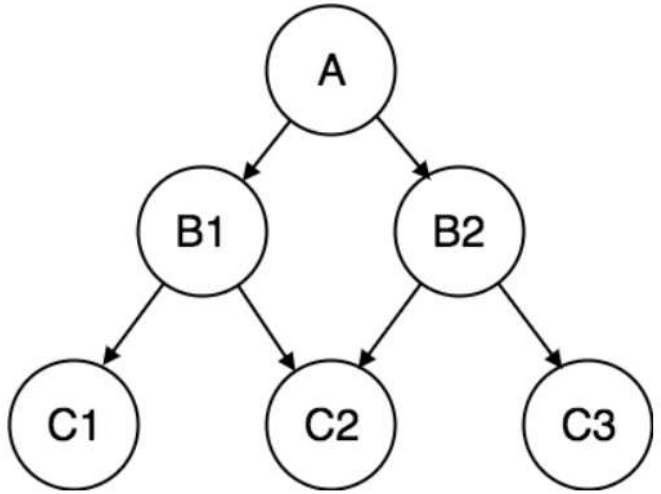
- This database model organises data into a **tree-like-structure**, with a single root, to which all the other data is linked. The hierarchy starts from the **Root** data, and expands like a tree, adding child nodes to the parent nodes.
- Hierarchical data model is the **oldest type of the data model**. It was developed by **IBM in 1968**. It organizes data in the tree-like structure.
- In this model, a child **node will only have a single parent node.**
- This model efficiently describes many real-world relationships like index of a book, recipes etc.
- In hierarchical model, data is organised into tree-like structure with one **one-to-many relationship between two different types of data**, for example, one department can have many courses, many professors and of-course many students.



Network Data Models

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





Relational Model

- In this model, data is organised in two-dimensional **tables** and the relationship is maintained by storing a common field.
- This model was introduced by **E.F Codd in 1970,** and since then it has been the most widely used database model
- we can say the only database model used around the world.

- The basic structure of data in the relational model is tables. All the information related to a particular type is stored in rows of that table.
- Hence, tables are also known as **relations** in relational model.
- In the coming tutorials we will learn how to design tables, normalize them to reduce data redundancy and how to use Structured Query language to access data from tables.

student_id	name	age
1	Akon	17
2	Bkon	18
3	Ckon	17
4	Dkon	18

subject_id	name	teacher
1	Java	Mr. J
2	C++	Miss C
3	C#	Mr. C Hash
4	Php	Mr. P H P

student_id	subject_id	marks
1	1	98
1	2	78
2	1	76
3	2	88

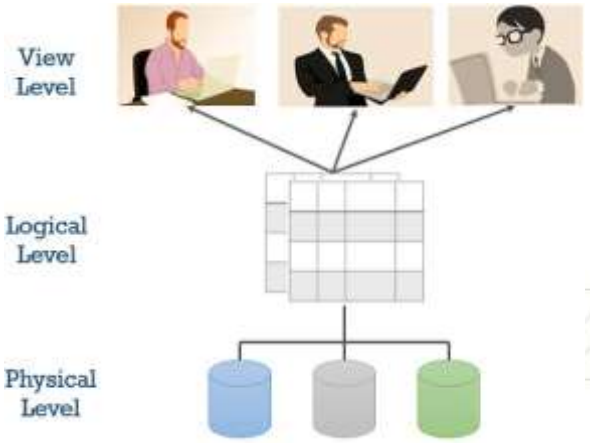
View of Data in DBMS

VIPS Technical Campus — SCHOOL OF ENGINEERING AND TECHNOLOGY

View of data

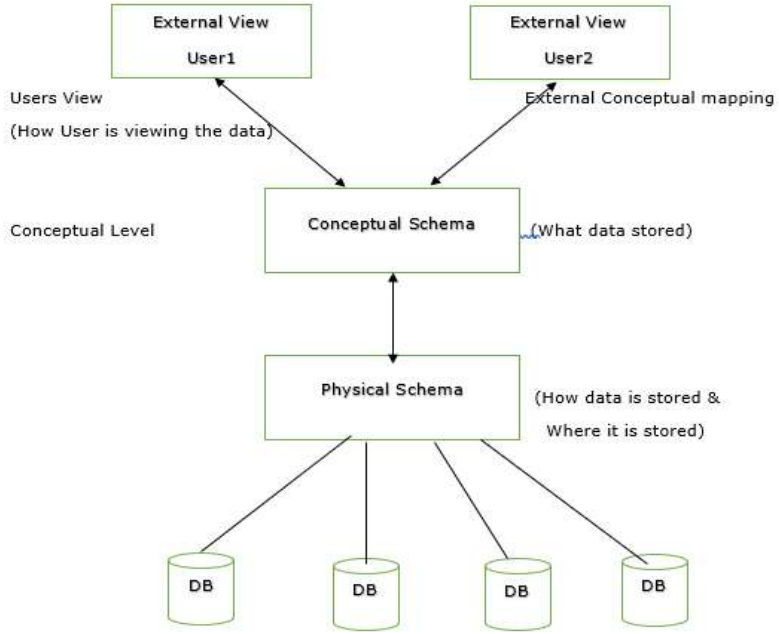
How the data is **visualized** at each level of data abstraction?

Data abstraction allow developers to keep complex data structures **away** from the users. The developers achieve this by hiding the complex data structures through **levels of abstraction.** The **three level database architecture** allows a clear separation of the view from the **external** data representation and from the **physical** data structure layout .



Data Abstraction

Data abstraction is **hiding the complex data structure** in order to **simplify the user's interface** of the system. It is done because many of the users interacting with the database system are not that much computer trained to understand the complex data structures of the database system.

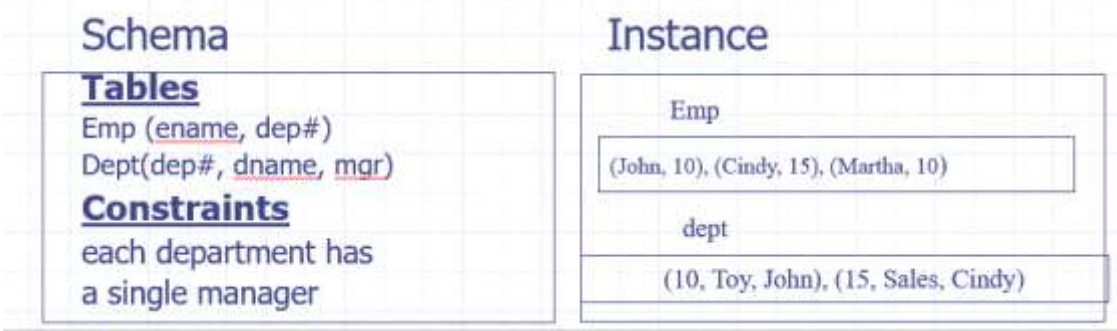


- **Design of a database is called the schema.**
- Schema is of **three types**: **Physical schema, logical schema and view schema**.
- The data stored in database at a particular moment of time is called **instance of database**.
- Database schema defines the variable declarations in tables that belong to a particular database; the value of these variables at a moment of time is called the instance of that database.

Instances and Schemas

- Similar to types and variables in programming languages
- **Schema** – the logical structure of the database
  - Example: The database consists of information about a set of customers and accounts and the relationship between them)
  - Analogous to type information of a variable in a program
  - **Physical schema**: database design at the physical level
  - **Logical schema**: database design at the logical level
  - **View Schema:**
- **Instance** – the actual content of the database at a particular point in time
  - Analogous to the value of a variable
- **Physical Data Independence** – the ability to modify the physical schema without changing the logical schema
  - Applications depend on the logical schema
  - In general, the interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.

Example: Database schema defines the variable declarations in tables that belong to a particular database; the value of these variables at a moment of time is called the instance of that database



Levels of Abstraction

- **Physical level:** describes how a record (e.g., customer) is stored. This is the lowest level in the three level architecture. It is also known as the internal level. The physical level describes how data is actually stored in the database. In the lowest level, this data is stored in the external hard drives in the form of bits and at a little high level, it can be said that the data is stored in files and folders. The physical level also discusses compression and encryption techniques.

- **Logical level:** describes data stored in database, and the relationships among the data. It describes how the database appears to the users conceptually and the relationships between various data tables. The conceptual level does not care for how the data in the database is actually stored.

type *customer* = **record**
customer_id : string;
customer_name : string;
customer_street : string;
customer_city : string;
end;

- **View level:** application programs hide details of data types. Views can also hide information (such as an employee's salary) for security purposes. This is the highest level in the three level architecture and closest to the user. It is also known as the view level. The external level only shows the relevant database content to the users in the form of views and hides the rest of the data. So different users can see the database as a different view as per their individual requirements.

Three-Schema Architecture:

The main objective of this architecture is to have an effective separation between the **user interface** and the **physical database**. So, the user never has to be concerned regarding the internal storage of the database and it has a simplified interaction with the database system.

The three-schema architecture defines the view of data at three levels:

1.Physical level (internal level)

2.Logical level (conceptual level)

3.View level (external level)

1. Physical Level/ Internal Level

The physical or the internal level schema describes **how the data is stored in the hardware**. It also describes how the data can be accessed. The physical level shows the data abstraction at the lowest level and it has **complex data structures**. Only the database administrator operates at this level.

2. Logical Level/ Conceptual Level

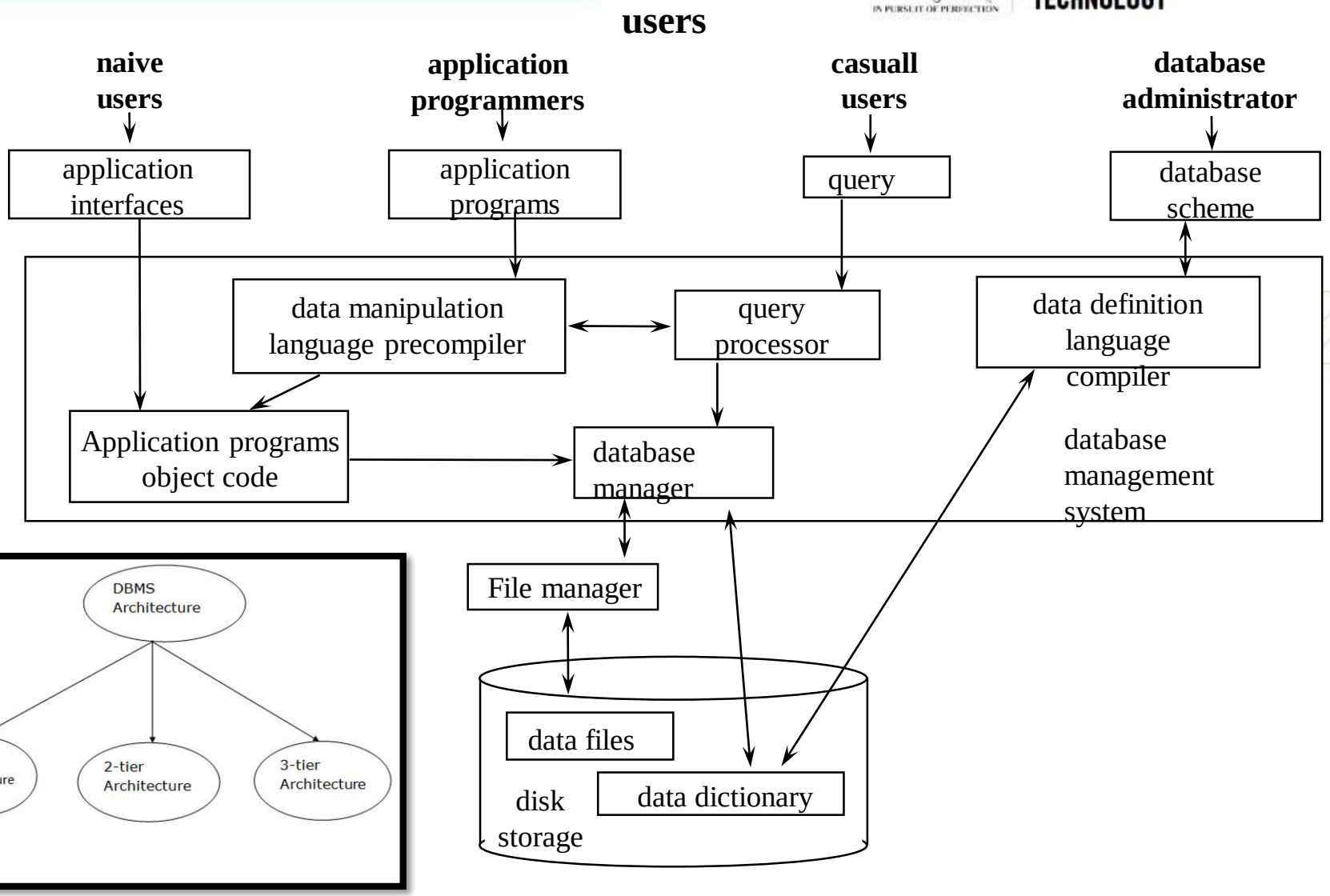
Here, the data is stored in the form of the **entity set**, **entities**, their **data types**, the **relationship** among the entity sets, **user operations** performed to retrieve or modify the data and certain **constraints on the data**. Well adding constraints to the view of data adds the security. As users are restricted to access some particular parts of the database. It is the developer and database administrator who operates at the logical or the conceptual level.

3. View Level/ User level/ External level

It is the highest level of data abstraction and exhibits only a part of the whole database. It exhibits the data in which the user is interested. The view level can describe many views of the same data. Here, the user retrieves the information using different application from the database.

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.



1-Tier Architecture

•In this architecture, the database is directly available to the user. It means the user can directly sit on the DBMS and uses 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.

•**1 Tier Architecture** in DBMS is the simplest architecture of Database in which the client, server, and Database all reside on the same machine. A simple one tier architecture example would be anytime you install a Database in your system and access it to practice SQL queries. But such architecture is rarely used in production.

Single Tier Architecture

2-Tier Architecture

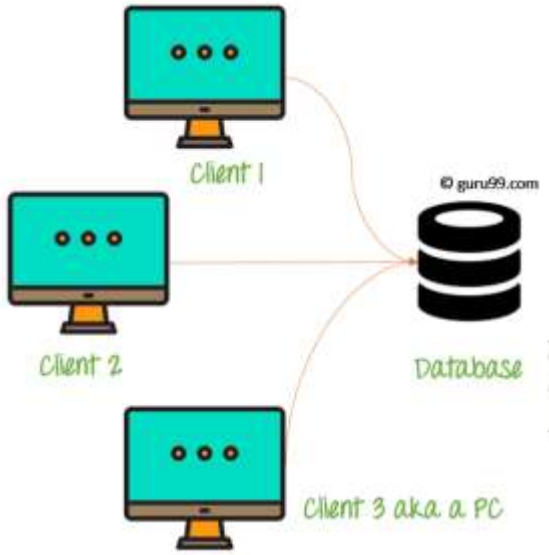
The 2-Tier architecture is same as basic client-server. In the two-tier architecture, applications on the client end can directly communicate with the database at the server side. For this interaction, API's like: **ODBC**, **JDBC** are used.

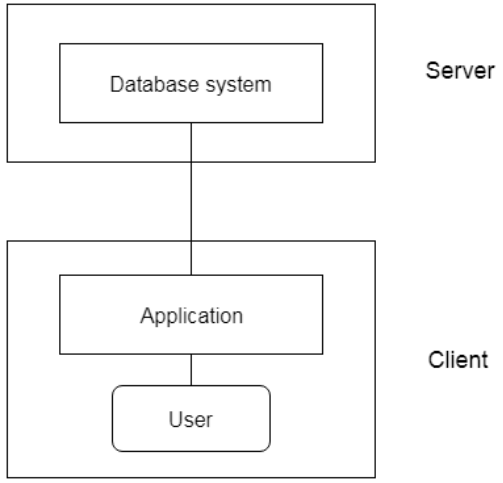
•The user interfaces and application programs are run on the client-side.

•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.

•A **2 Tier Architecture** in DBMS is a Database architecture where the presentation layer runs on a client (PC, Mobile, Tablet, etc.), and data is stored on a server called the second tier. Two tier architecture provides added security to the DBMS as it is not exposed to the end-user directly. It also provides direct and faster communication.





3-Tier Architecture

A **3 Tier Architecture** in DBMS is the most popular client server architecture in DBMS in which the development and maintenance of functional processes, logic, data access, data storage, and user interface is done independently as separate modules. Three Tier architecture contains a presentation layer, an application layer, and a database server.

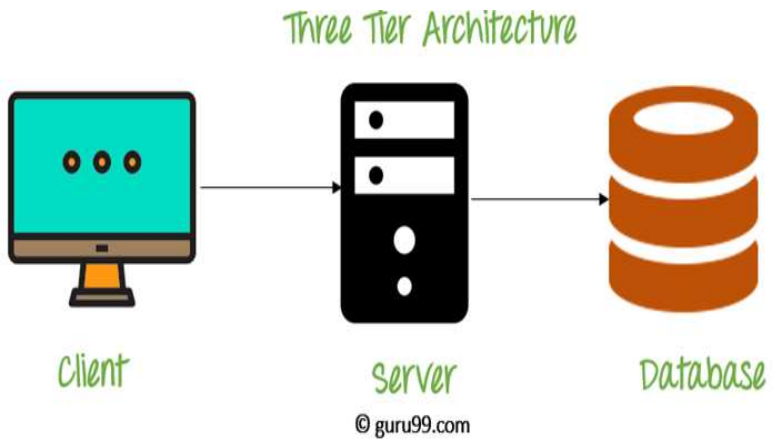
3-Tier database Architecture design is an extension of the 2-tier client-server architecture. A 3-tier architecture has the following layers:

1.Presentation layer (your PC, Tablet, Mobile, etc.)

2.Application layer (server)

3.Database Server

The Application layer resides between the user and the DBMS, which is responsible for communicating the user's request to the DBMS system and send the response from the DBMS to the user. The application layer(business logic layer) also processes functional logic, constraint, and rules before passing data to the user or down to the DBMS.



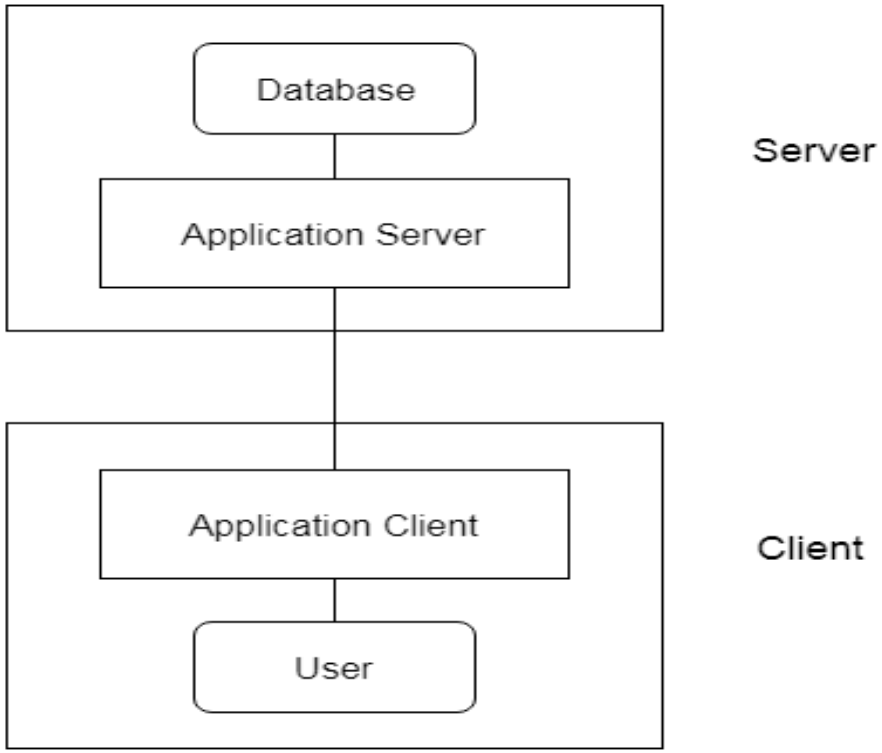
3-Tier Architecture

•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 application.



Components of DBMS

- A database environment is a collection of components that regulates the use of data, management, and a group of data.
- These components consist of people, the technique of Handel the database, data, hardware, software, etc

1. Hardware

•In DBMS, the hardware includes output devices like a printer, monitor, etc., and storage devices like a hard disk.

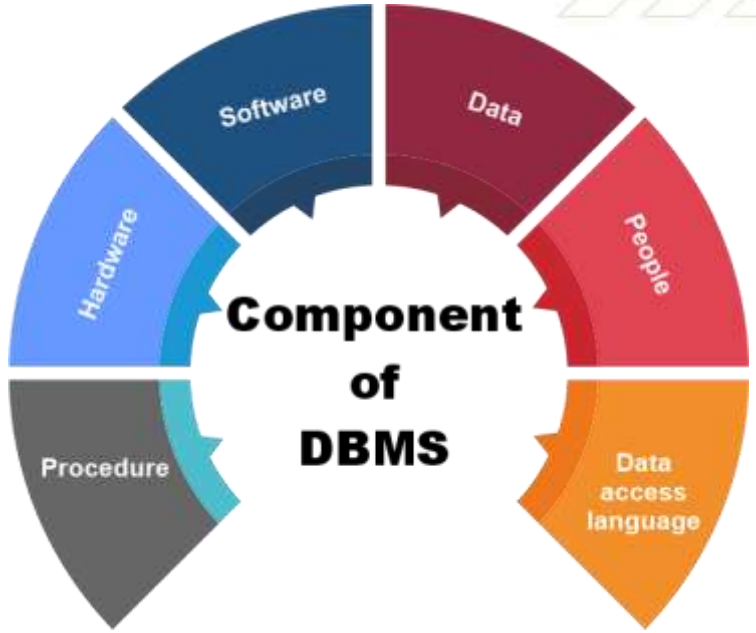
•In DBMS, information hardware is the most important visible part. The equipment which is used for the visibility of the data is the printer, computer, scanner, etc.

•With the help of hardware, the DBMS can access and update the database.

•The server can store a large amount of data, which can be shared with the help of the user's own system.

•The database can be run in any system that ranges from microcomputers to mainframe computers. And this database also provides an interface between the real worlds to the database.

•When we try to run any database software like MySQL, we can type any commands with the help of our keyboards, and RAM, ROM, and processor are part of our computer system.



2. Software

•Software is the main component of the DBMS.

•Software is defined as the collection of programs that are used to instruct the computer about its work. The software consists of a set of procedures, programs, and routines associated with the computer system's operation and performance. Also, we can say that computer software is a set of instructions that is used to instruct the computer hardware for the operation of the computers.

•The software includes so many software like network software and operating software. The database software is used to access the database, and the database application performs the task.

•This software has the ability to understand the database accessing language and then convert these languages to real database commands and then execute the database.

•This is the main component as the total database operation works on a software or application. We can also be called as database software the wrapper of the whole physical database, which provides an easy interface for the user to store, update and delete the data from the database.

•Some examples of DBMS software include MySQL, Oracle, SQL Server, dBase, FileMaker, Clipper, Foxpro, Microsoft Access, etc.

3. Data

•The term data means the collection of any raw fact stored in the database. Here the data are any type of raw material from which meaningful information is generated.
•The database can store any form of data, such as structural data, non-structural data, and logical data.
•The structured data are highly specific in the database and have a structured format. But in the case of non-structural data, it is a collection of different types of data, and these data are stored in their native format.
•We also call the database the structure of the DBMS. With the help of the database, we can create and construct the DBMS. After the creation of the database, we can create, access, and update that database.
•The main reason behind discovering the database is to create and manage the data within the database.
•Data is the most important part of the DBMS. Here the database contains the actual data and metadata. Here metadata means data about data.
•For example, when the user stores the data in a database, some data, such as the size of the data, the name of the data, and some data related to the user, are stored within the database. These data are called metadata.

4. Procedures

•The procedure is a type of general instruction or guidelines for the use of DBMS. This instruction includes how to set up the database, how to install the database, how to log in and log out of the database, how to manage the database, how to take a backup of the database, and how to generate the report of the database.
•In DBMS, with the help of procedure, we can validate the data, control the access and reduce the traffic between the server and the clients. The DBMS can offer better performance to extensive or complex business logic when the user follows all the procedures correctly.
•The main purpose of the procedure is to guide the user during the management and operation of the database.
•The procedure of the databases is so similar to the function of the database. The major difference between the database procedure and database function is that the database function acts the same as the SQL statement. In contrast, the database procedure is invoked using the CALL statement of the DBMS.
•Database procedures can be created in two ways in enterprise architecture. These two ways are as below.

5. Database Access Language

•Database Access Language is a simple language that allows users to write commands to perform the desired operations on the data that is stored in the database.

•Database Access Language is a language used to write commands to access, upsert, and delete data stored in a database.

•Users can write commands or query the database using Database Access Language before submitting them to the database for execution.

•Through utilizing the language, users can create new databases and tables, insert data and delete data.

•Examples of database languages are SQL (structured query language), My Access, Oracle, etc. A database language is comprised of two languages.

1. **Data Definition Language(DDL):**
2. **DML**

People

•The people who control and manage the databases and perform different types of operations on the database in the DBMS.

•The people include database administrator, software developer, and End-user.

•Database administrator-database administrator is the one who manages the complete database management system. DBA takes care of the security of the DBMS, its availability, managing the license keys, managing user accounts and access, etc.

•Software developer- theThis user group is involved in developing and designing the parts of DBMS. They can handle massive quantities of data, modify and edit databases, design and develop new databases, and troubleshoot database issues.

•End user - These days, all modern web or mobile applications store user data. How do you think they do it? Yes, applications are programmed in such a way that they collect user data and store the data on a DBMS system running on their server. End users are the ones who store, retrieve, update and delete data.

•The users of the database can be classified into different groups.

- Native Users, Online Users, Sophisticated Users, Specialized Users, Application Users
- DBA - Database Administrator

Database Users and Administrators

Database Users and Administrators

- A primary goal of a database system is to retrieve information from and store new information in the database.
- People who work with a database can be categorized as database users or database administrators.

Users of Database

- Depending on the degree of expertise or mode of interaction with DBMS the users of database
- Database Administrator
- Application Programmers
- Sophisticated Users
- Naive Users

Database Administrator (DBA)

- Database Administrator (DBA) is a person/team who defines the schema and also controls the 3 levels of database.
- The DBA will then create a new account id and password for the user if he/she need to access the data base.
- DBA is also responsible for providing security to the data base and he allows only the authorized users to access/modify the data.
- DBA also monitors the recovery and back up and provide technical support.
- The DBA has a DBA account in the DBMS which called a system or superuser account.
- DBA repairs damage caused due to hardware and/or software failures.

Application Programmer

- Application Programmer are the back end programmers who writes the code for the application programs.
- They are the computer professionals.
- These programs could be written in Programming languages such as Visual Basic, Developer, C, FORTRAN, COBOL etc.

Sophisticated users

- Sophisticated users can be engineers, scientists, business analyst, who are familiar with the database.
- They can develop their own data base applications according to their requirement.
- They don't write the program code but they interact the data base by writing SQL queries directly through the query processor.

Naive Users

- Parametric End Users are the unsophisticated who don't have any DBMS knowledge but they frequently use the data base applications in their daily life to get the desired results.
- For examples, Railway's ticket booking users are naive users.
- Clerks in any bank is a naive user because they don't have any DBMS knowledge but they still use the database and perform their given task.

Data Independence

- The capacity to change the schema at one level of a database system without having to change the schema at the next higher level.
- Two types of data independence: Logical data independence, Physical data independence

Logical data independence

- Logical data independence is the capacity to change the conceptual schema without having to change external schemas or application programs.

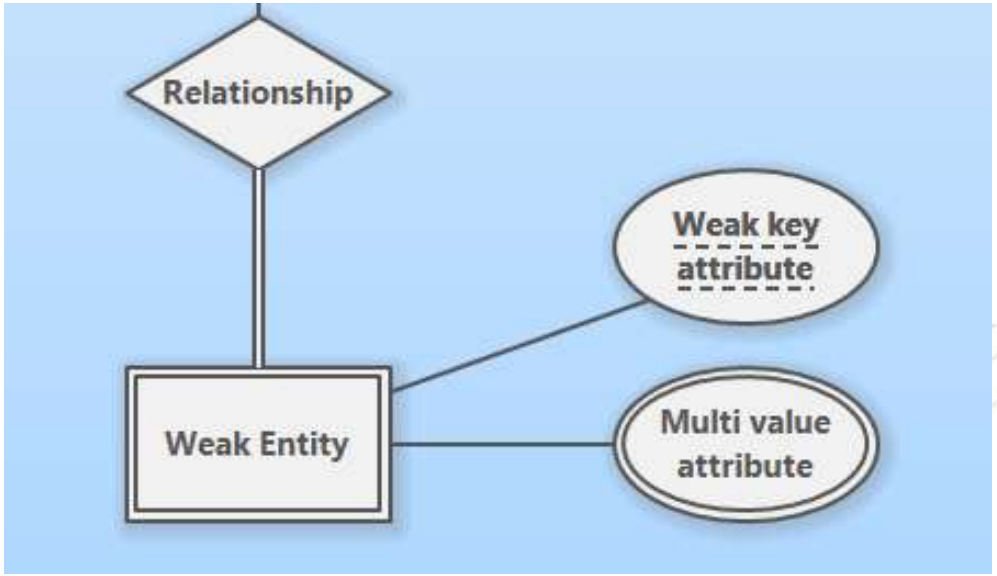
Physical data independence

- Physical data independence is the capacity to change the internal schema without having to change the conceptual schema.
- Data independence occurs because when the schema is changed at some level, the schema at the next higher level remains unchanged; only the mapping between the two levels is changed.

ER Model

ER Diagram

- It is used for conceptual data design of database.
- Collection of **entities** and their properties called **attributes** and **relationship** between them
- Diagrammatic representation and easy to understand.
- Entity Relationship Diagram (ER Diagram) pictorially explains the relationship between entities to be stored in a database.
- the ER Diagram is a **structural design of the database.**
- It acts as a **framework** created with specialized symbols for the purpose of defining the relationship between the database entities.
- ER diagram is created based on three principal components: **entities, attributes, and relationships.**



Entity Name

Entity

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

Example: Car, Student

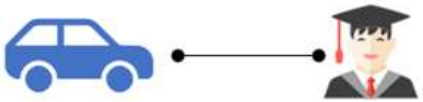
Jack

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

Entity

- An entity is an **object or component of data**. An **entity is represented as rectangle in an ER diagram.**

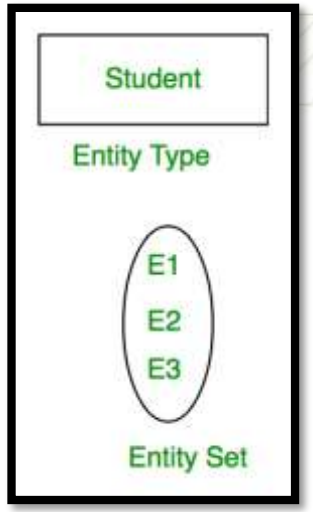
Student Teacher Projects

For example: In the following ER diagram we have two entities **Student** and **College** and these two entities have many to one relationship as many students study in a single college.

Entity, Entity Type, Entity Set –

An Entity may be an **object with a physical existence** – a particular person, car, house, or employee – or it may be an object with a conceptual existence – a company, a job, or a university course.

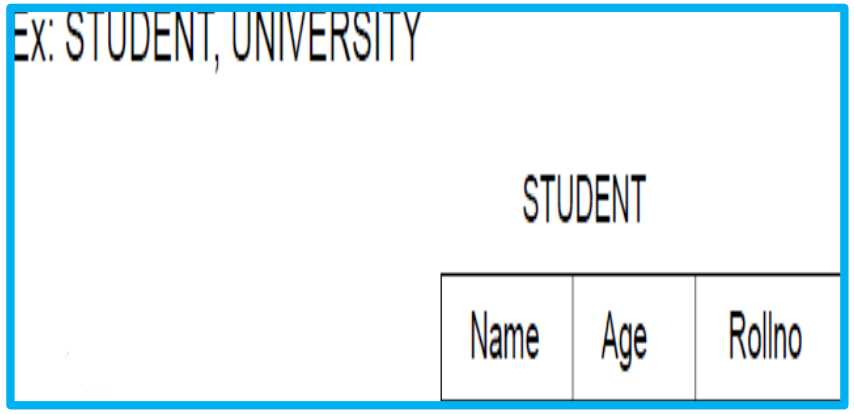
An Entity is an **object of Entity Type** and a set of all entities is called as an entity set. e.g.; E1 is an entity having Entity Type Student and set of all students is called Entity Set. In ER diagram, Entity Type is represented as:

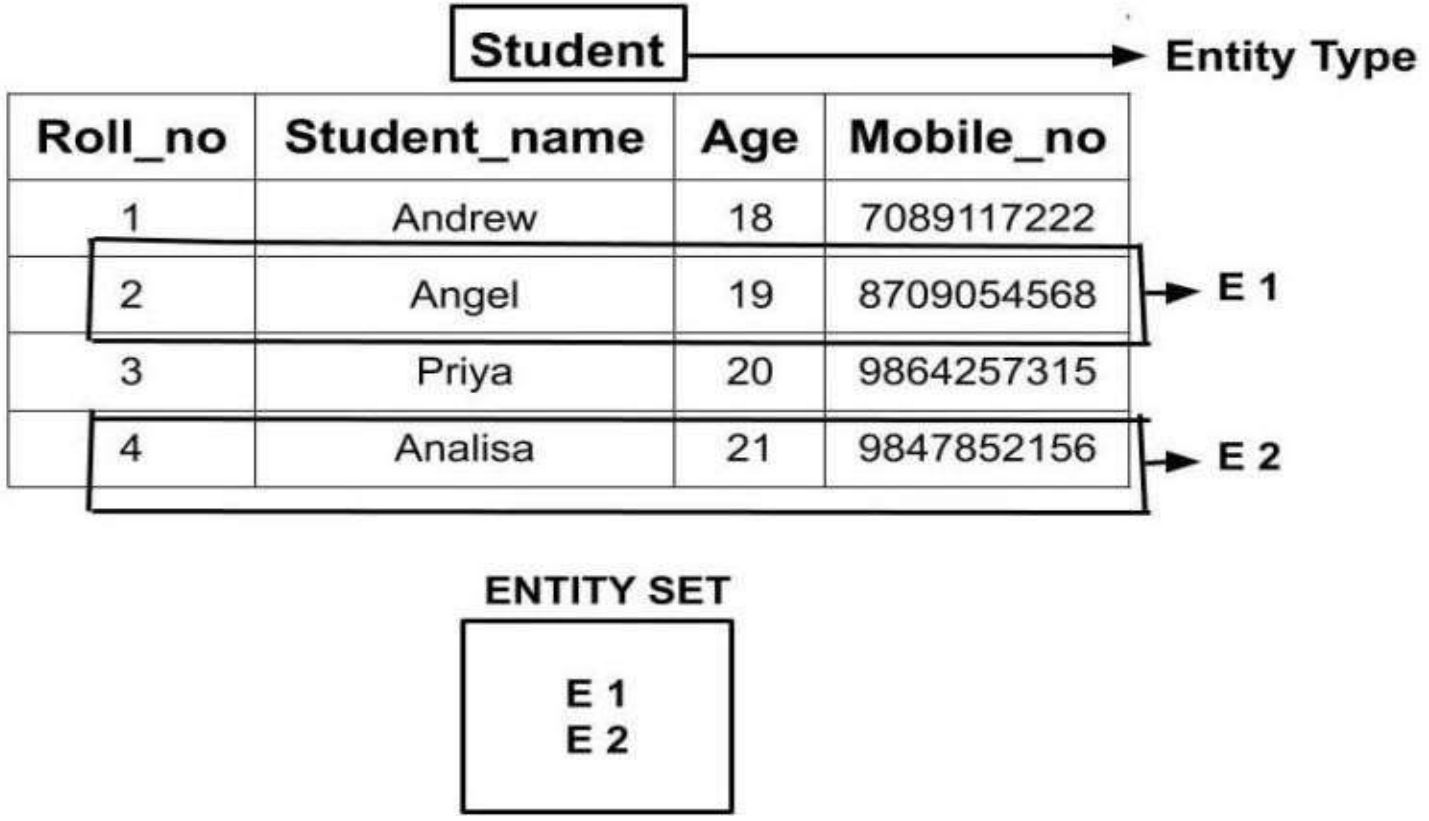


Weak Entity: An entity that cannot be uniquely identified by its own attributes and relies on the relationship with other entity is called weak entity. The weak entity is represented by a double rectangle.

Entity Type

- The entity type is a collection of the entity having similar attributes.
- an **entity type** in an ER diagram is defined by a name and a set of attributes
- *We use a rectangle to represent an entity type in the E-R diagram, not entity.*





Entity set

- The collection of same type of entities that is their attributes are same is called entity set.

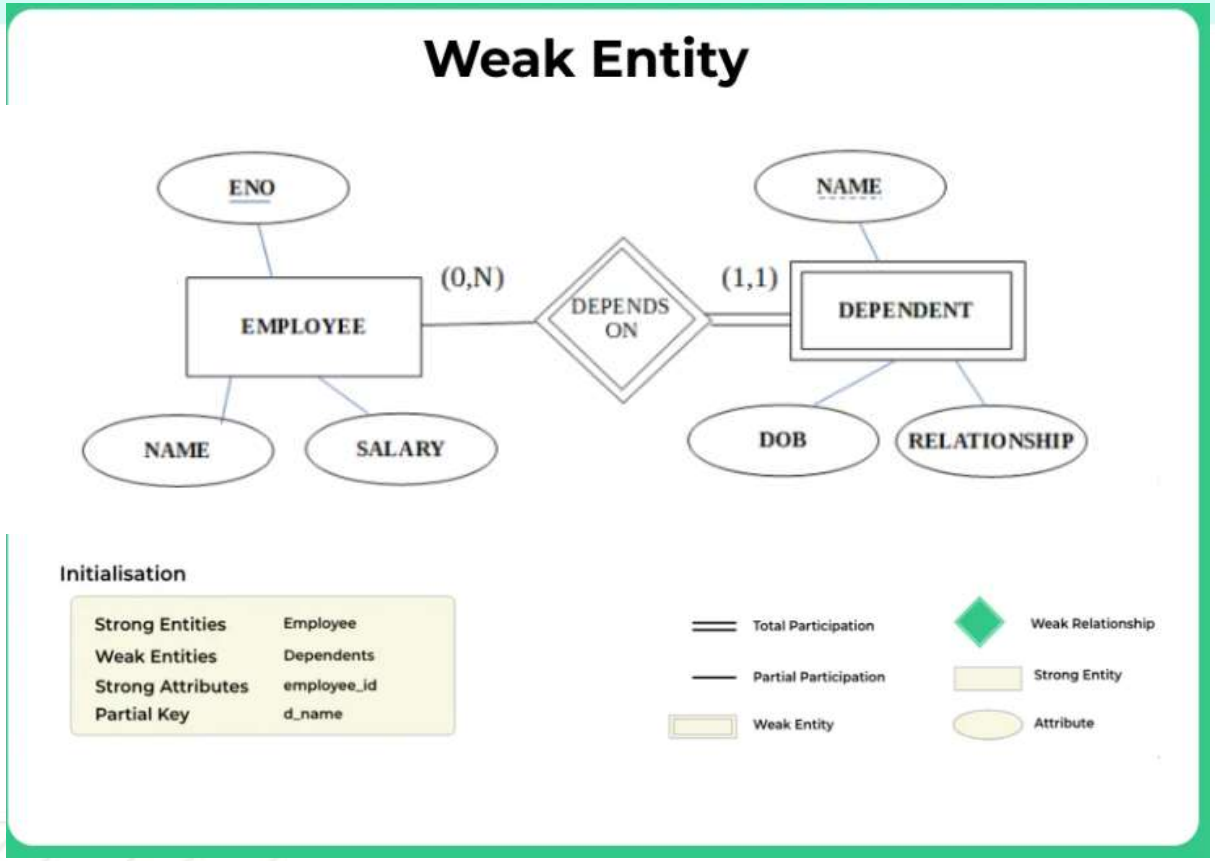
TYPES OF ENTITY TYPES

Strong entity type

- Entity types that have at least one key attribute.
- A strong entity is not dependent of any other entity in the schema.
- A strong entity will always have a primary key.
- Strong entities are represented by a single rectangle.
- The relationship of two strong entities is represented by a single diamond.

- **Weak entity type**
  - Entity type that does not have any key attribute.
  - A weak entity is dependent on a strong entity to ensure the its existence.
  - Unlike a strong entity, a weak entity **does not have any primary key.**
  - It instead has a **partial discriminator key.**
  - A weak entity is represented by a **double rectangle.** The relation between one strong and one weak entity is represented by a **double diamond.**

Weak entity



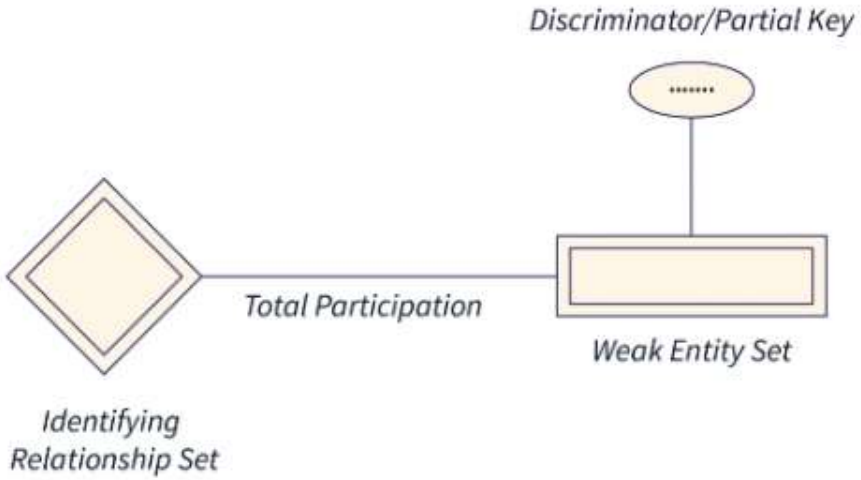
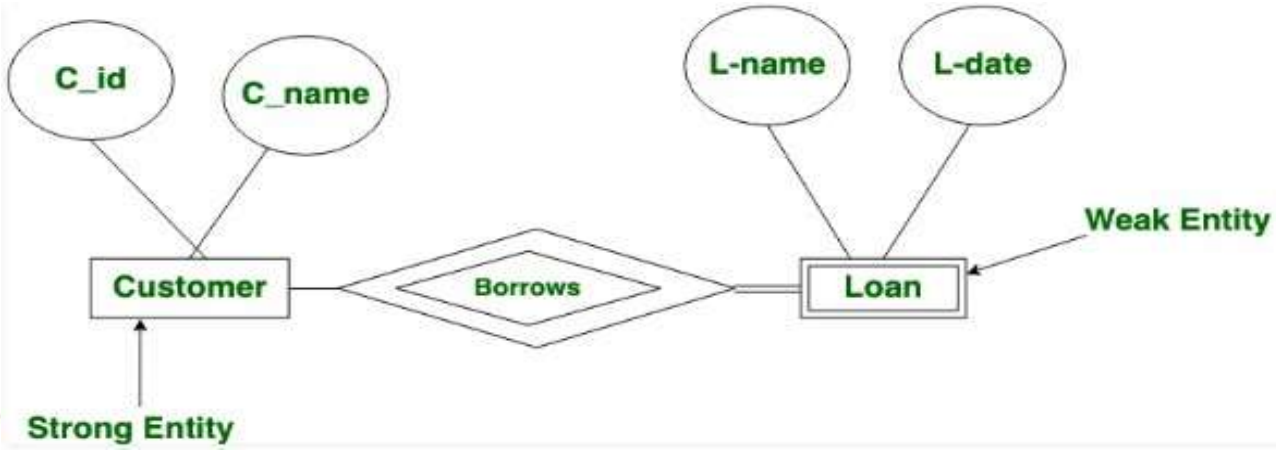


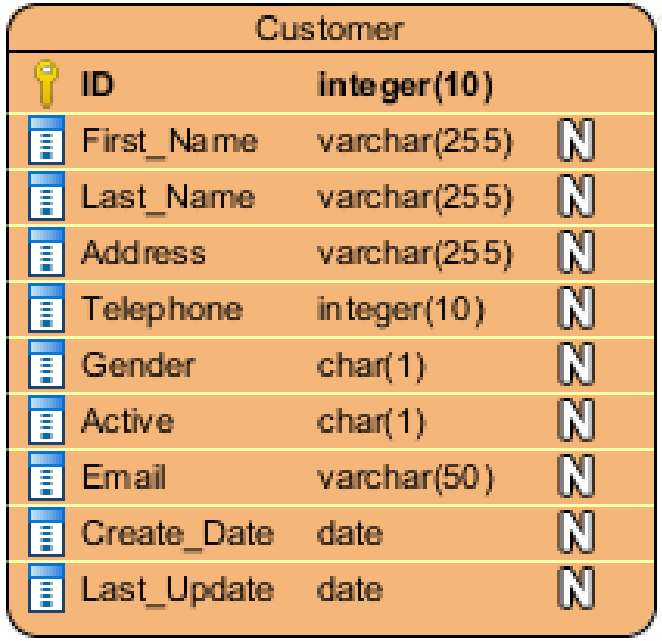
Fig: Representation of Weak Entity Set



Attributes

- Attributes are the **properties that define the entity type**.

- Attributes describes characteristics of entity
- Suppose we have a entity EMPLOYEE and its attributes are ENO, ESAL, ENAME etc..
- Attributes have some set of allowed or permitted values called Domain
- Attributes are represented by **OVAL**
- Each attribute of an entity set is associated with domain that means the set of values that can be assigned to that attribute for an entity.
- Also known as a column, an attribute is a **property or characteristic of the entity that holds it**.

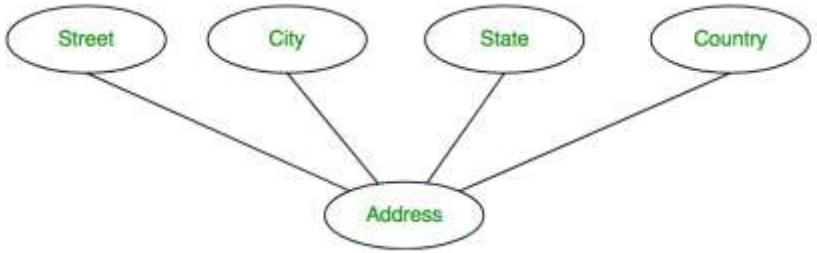


1. Key Attribute –

The attribute which **uniquely identifies each entity** in the entity set is called key attribute.For example, Roll_No will be unique for each student. In ER diagram, key attribute is represented by an oval with underlying lines.

2. Composite Attribute –

An attribute **composed of many other attribute** is called as composite attribute. For example, Address attribute of student Entity type consists of Street, City, State, and Country. In ER diagram, composite attribute is represented by an oval comprising of ovals.



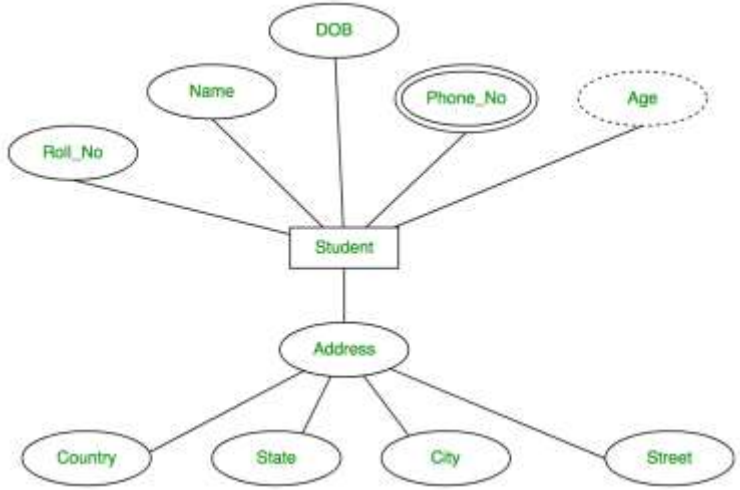
3. Multivalued Attribute –

An attribute consisting **more than one value** for a given entity. For example, Phone_No (can be more than one for a given student). In ER diagram, a multivalued attribute is represented by a double oval.

4. Derived Attribute –

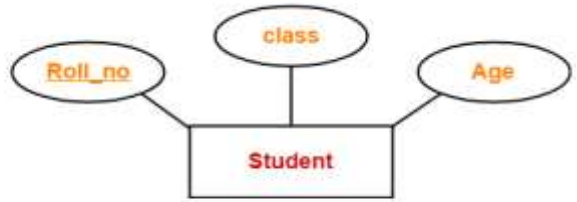
An attribute that can be **derived from other attributes** of the entity type is known as a derived attribute. e.g.; Age (can be derived from DOB). In ER diagram, the derived attribute is represented by a dashed oval. The **Age attribute** is hence called a derived attribute and is said to be derivable from the **DOB attribute**, which is called a **stored attribute .**

The complete entity type **Student** with its attributes can be represented as



Simple attribute vs Composite attribute

- Simple attributes
  - Attributes which are not divisible; that is they cannot be divided.
  - Eg: City, State, etc,.

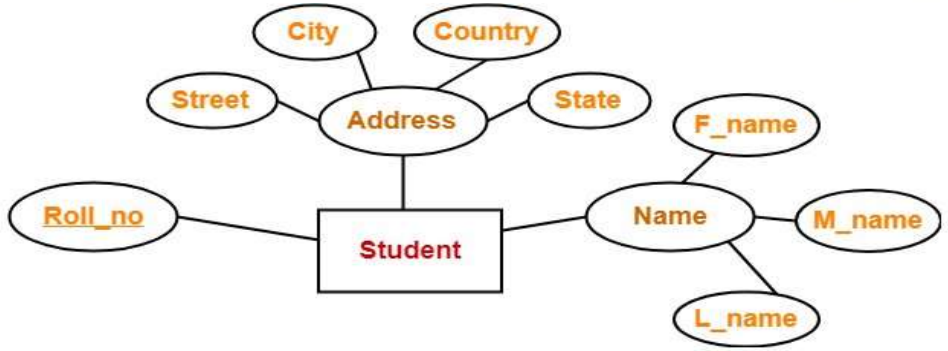


Here, all the attributes are simple attributes as they can not be divided further.

Domain of value set of an attribute

- Domain of an attribute is the allowed set of values of that attribute.
- Example: if attribute is 'grade', then its allowed values are A,B,C,F.
- Grade ={A, B,C,F}

- Composite Attribute
  - Attributes that can be divided into smaller sub parts
  - Example: Name attribute can be divided into FirstName, MiddleName, LastName

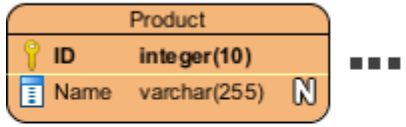


Here, the attributes "Name" and "Address" are composite attributes as they are composed of many other simple attributes.

Primary Key/ Key Attribute

Key Attribute –

The attribute which **uniquely identifies each entity** in the entity set is called key attribute. For example, Roll_No will be unique for each student. In ER diagram, key attribute is represented by an **oval with underlying lines.**



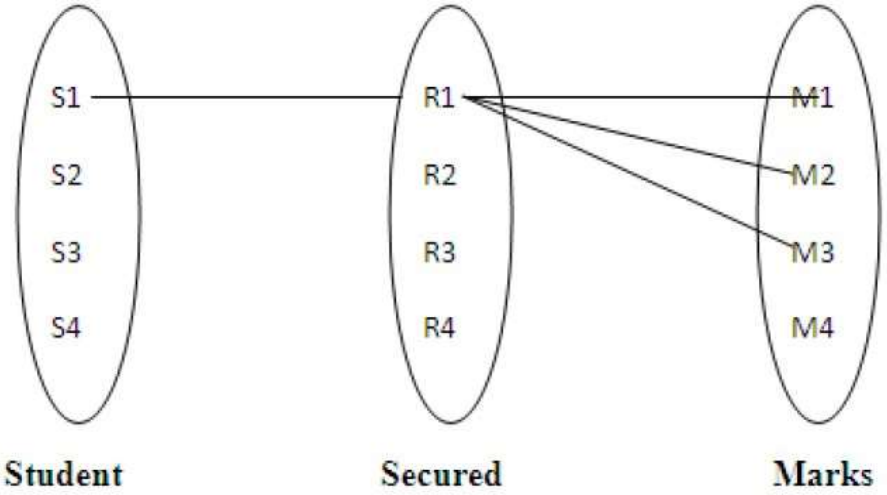
ID	Name
PDT-0001	Tiger T7 Bluetooth Headphones
PDT-0002	DD-027 In-Ear Headphones, Black
PDT-0002	SDB-21 Hi-Fi Stereo Over-ear Earphones
PDT-0003	Mr. 1022 Deep Bass Earbuds

Primary Key

Also known as PK, a primary key is a special kind of entity attribute that **uniquely defines a record in a database table**. In other words, there must not be two (or more) records that share the same value for the primary key attribute.

Relationship & Relationship type

- Relates two or more distinct entities with a specific meaning.
- It is an association between two or more entities of **same** or **different entity set**
  - For example, EMPLOYEE John works on the ProductX PROJECT or
  - EMPLOYEE Franklin manages the Research DEPARTMENT.



Relationship type: secured
Relationship set: {R1, R2, R3, R4}
Relationship instances: R1

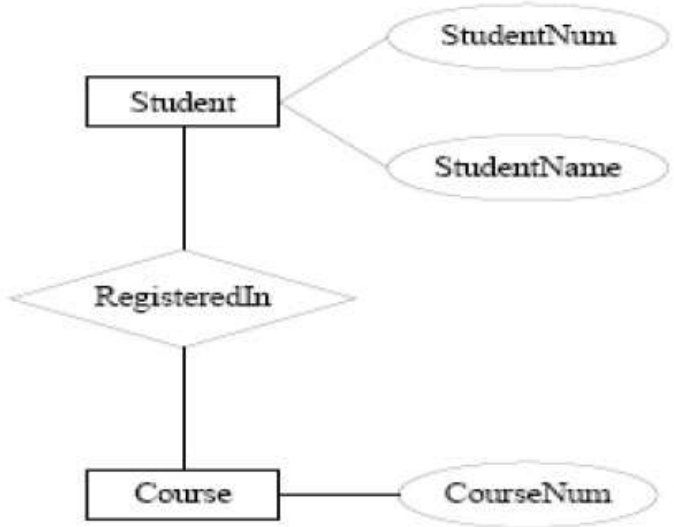
- A set of similar types of relationship

'Enrolled in' is a relationship that exists between entities **Student** and **Course**.

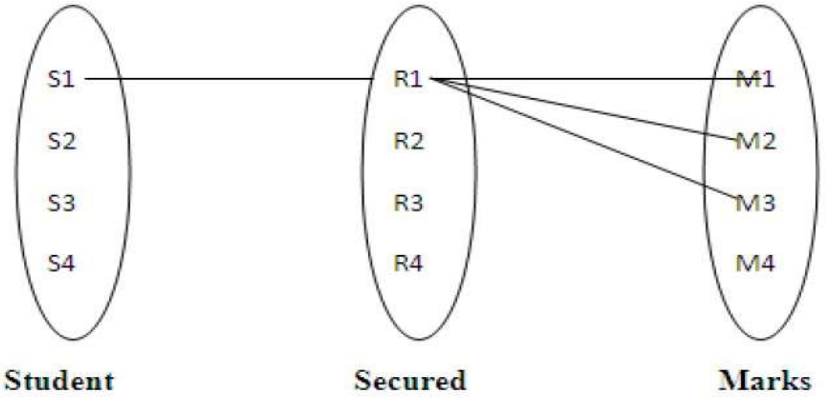
- Each relationship has
  - Name
  - Degree
  - Cardinality ratio

Relationship Set

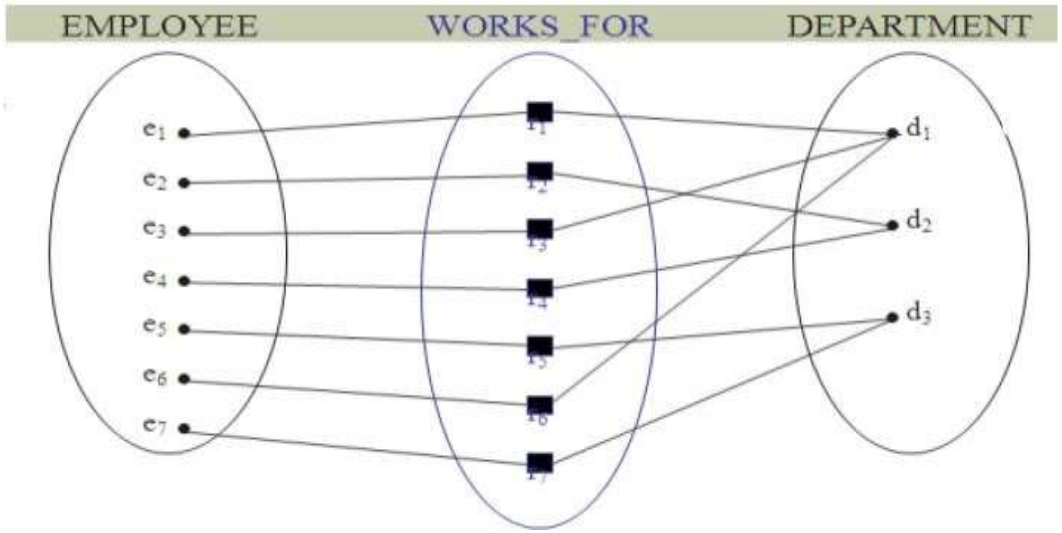
A **relationship set** is a set of relationships of the same type.

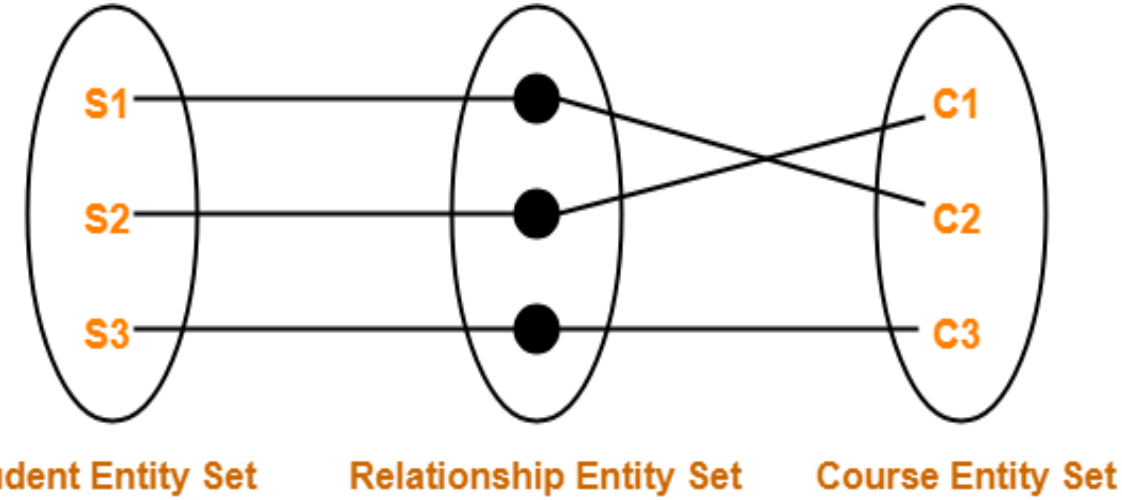


Graphical Representation of Relationship Sets



Relationship type: secured
Relationship set: {R1, R2, R3, R4}
Relationship instances: R1



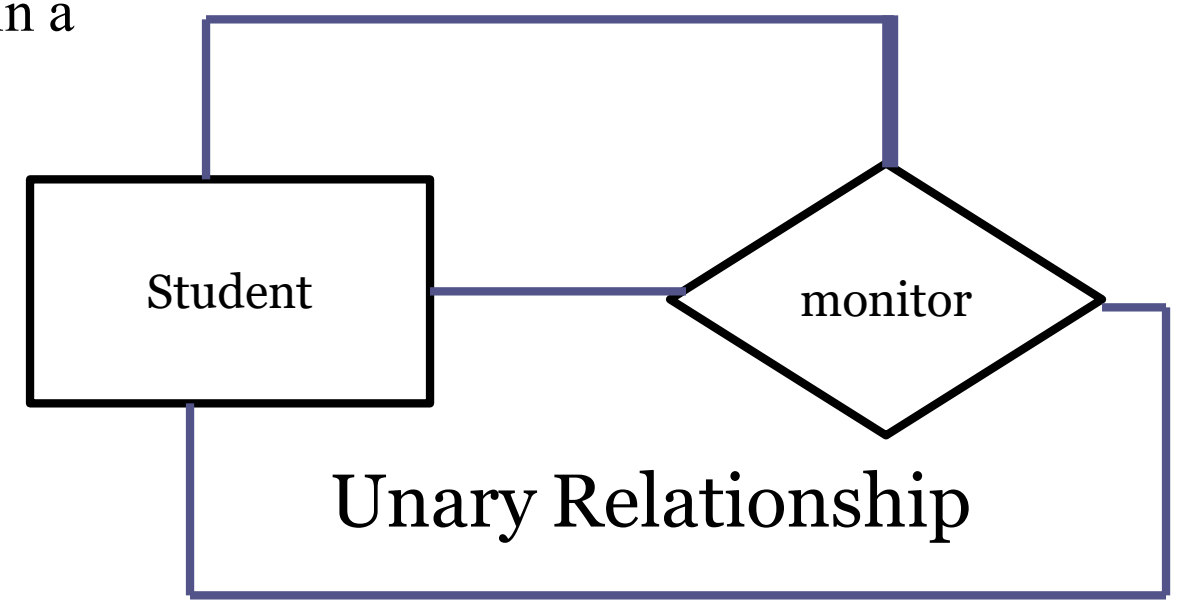


Set Representation of ER Diagram

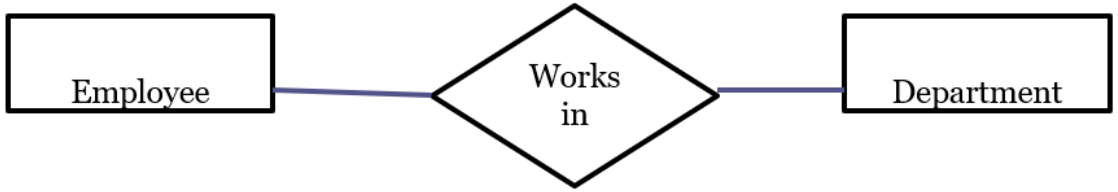
Degree of a relationship

It is the number of entity set which are participating in a relationship

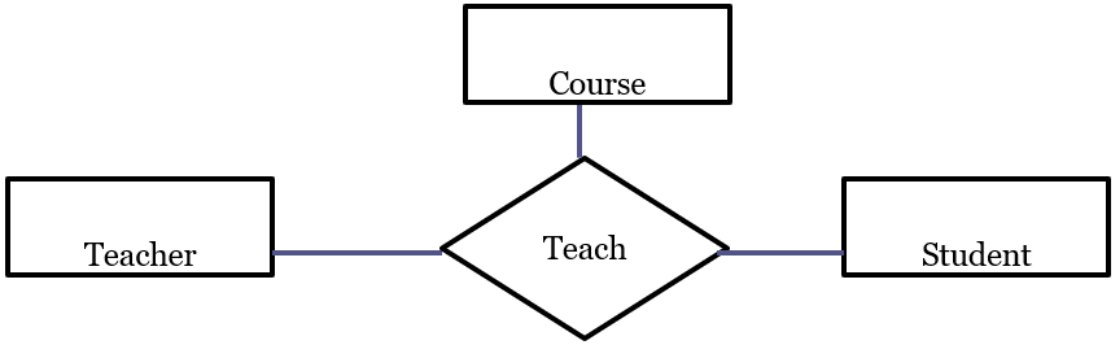
- **Unary relationship**
- **Binary Relationship**
- **Ternary Relationship**



- Binary Relationship

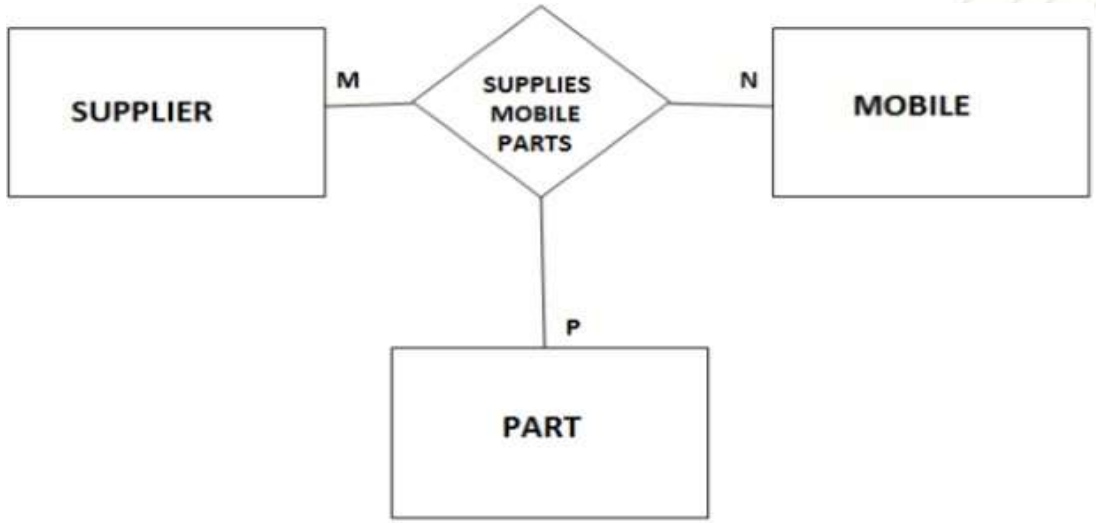


- Ternary Relationship



Relationship of degree 3

- In Ternary relationship three different Entities takes part in a Relationship.
- Relationship Degree = 3
- For Example: Consider a Mobile manufacture company. Three different entities involved:
  - Mobile - Manufactured by company.
  - Part - Mobile Part which company get from Supplier.
  - Supplier - Supplier supplies Mobile parts to Company.



- Mobile, Part and Supplier will participate simultaneously in a relationship. because of this fact when we consider cardinality we need to consider it in the context of two entities simultaneously relative to third entity.

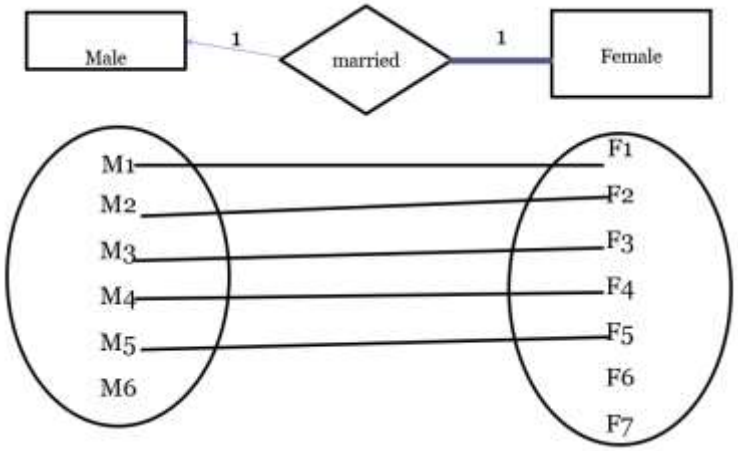
Cardinality Ratio (1)

- The cardinality ratio for a binary relationship specifies the maximum number of relationship instances to which an entity can take part in it. It also specifies number of entities to which other entity can be related by a relationship
- Types
  - **One-to-one (1:1)**
  - **One-to-many (1: N)**
  - **Many-to-one (N: 1)**
  - **Many-to-many (M: N)**

- We express cardinality ratio by drawing
- directed line (→), signifying "one," or an
- undirected line (—), signifying "many,"

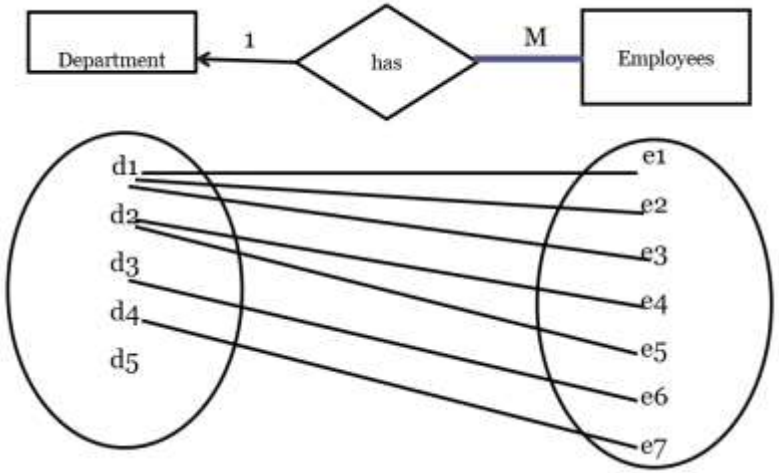
One to One(1:1)

- When only a **single instance** of an entity is **associated** with **single instance** of other entity by a relationship
- When every entity of one entity set is related to maximum one entity of other entity set.



One to Many (1:M)

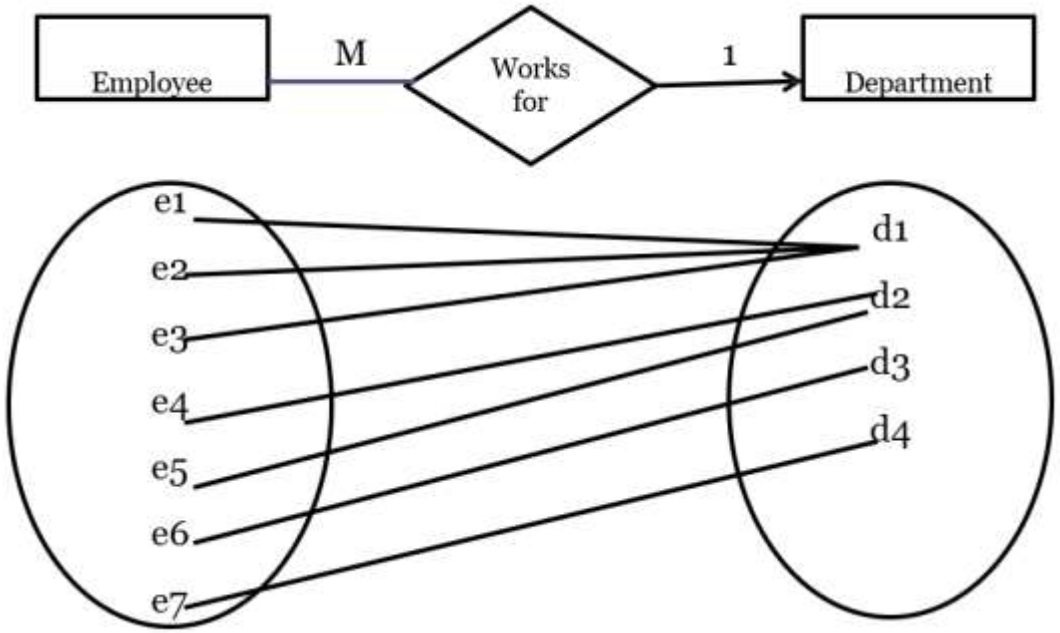
- When every entity of first entity set is related to at most (max) n entities of other entity set then it is one to many



Cardinality Ratio(2)

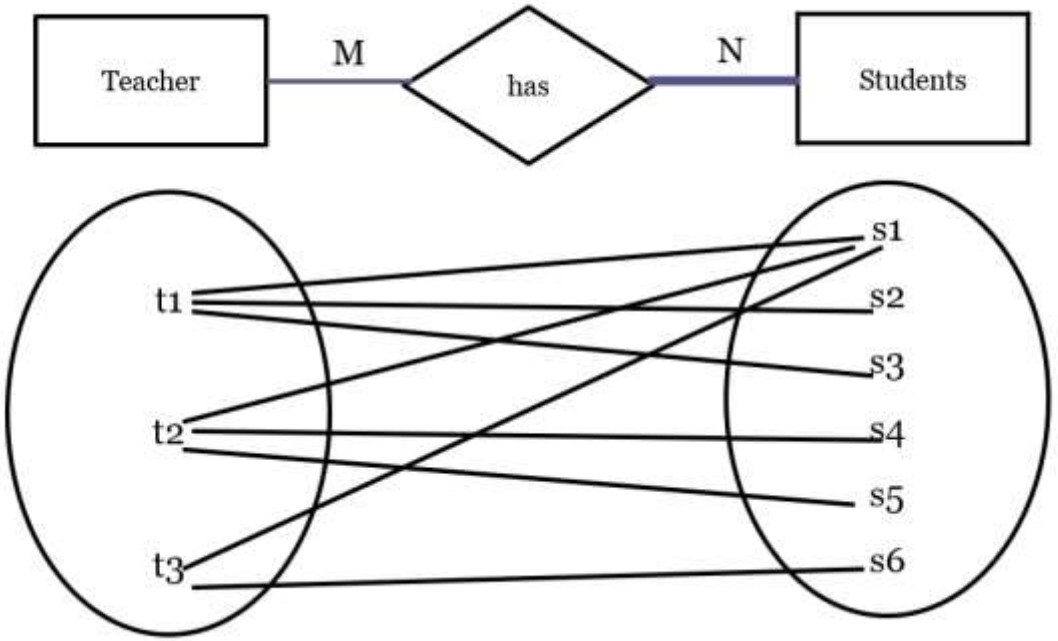
Many to One (M:1)

- When many entities of first entity set is related to 1 entity of other entity set then it is many to one



Many to Many(M:N)

- When many occurrences of one entity is related to many occurrences of another entity.



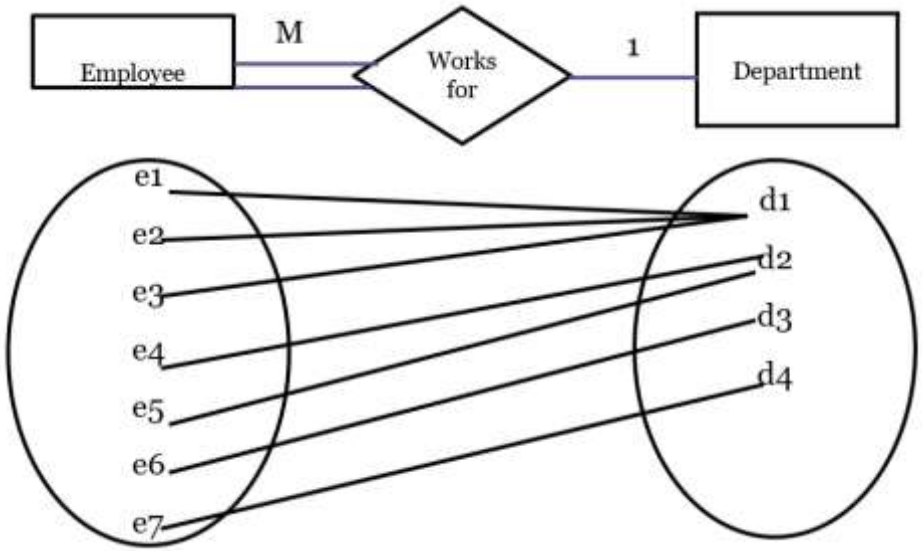
Participation constraints

- It specifies whether the existence of an entity depends on being related to another entity through relationship types.These constraints defines max and min
- Maximum cardinality:It defines maximum number of times an entity can participate in a relationship
- Minimum Cardinality:It defines minimum number of times an entity can participate in a relationship

Total participation

Every entity in the entity type participates in at least one relationship in the relationship type

- Represented by double lines
- Minimum and maximum cardinality represented as (m,n)



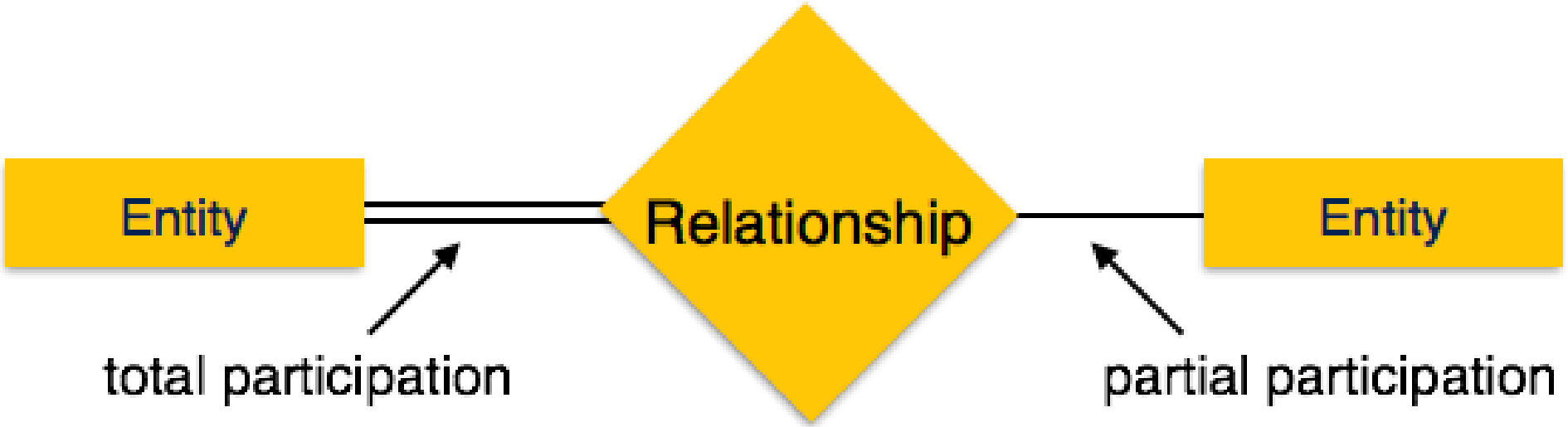
Partial participation

- Some entities may not participate in any relationship in the relationship type
- Represented by single line

Participation Constraints

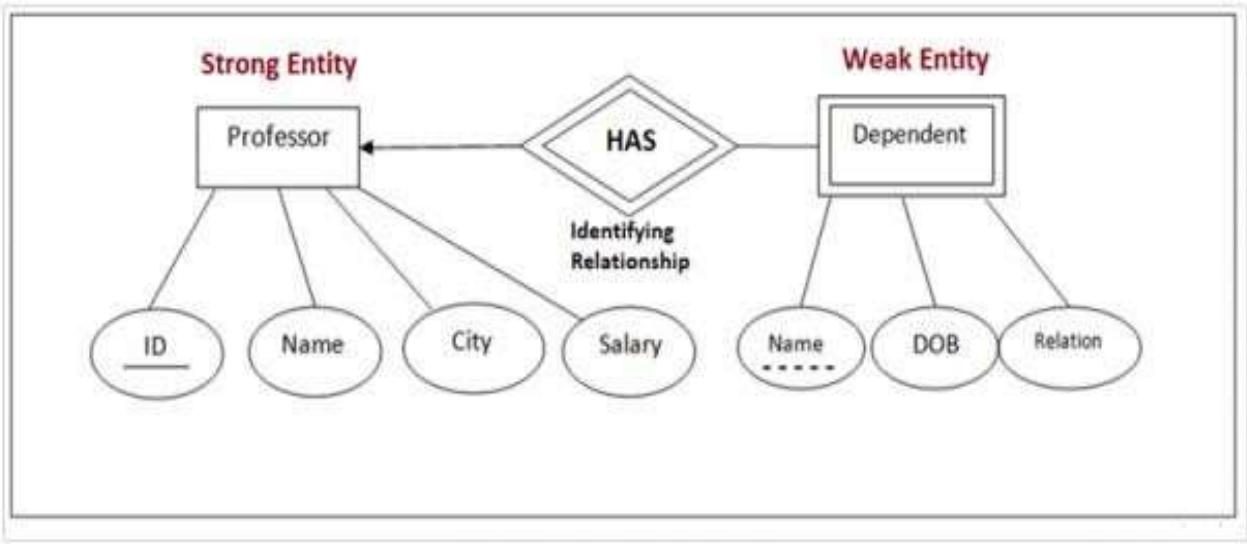
•**Total Participation** − Each entity is involved in the relationship. Total participation is represented by double lines.

•**Partial participation** − Not all entities are involved in the relationship. Partial participation is represented by single lines.



Identifying Relationship

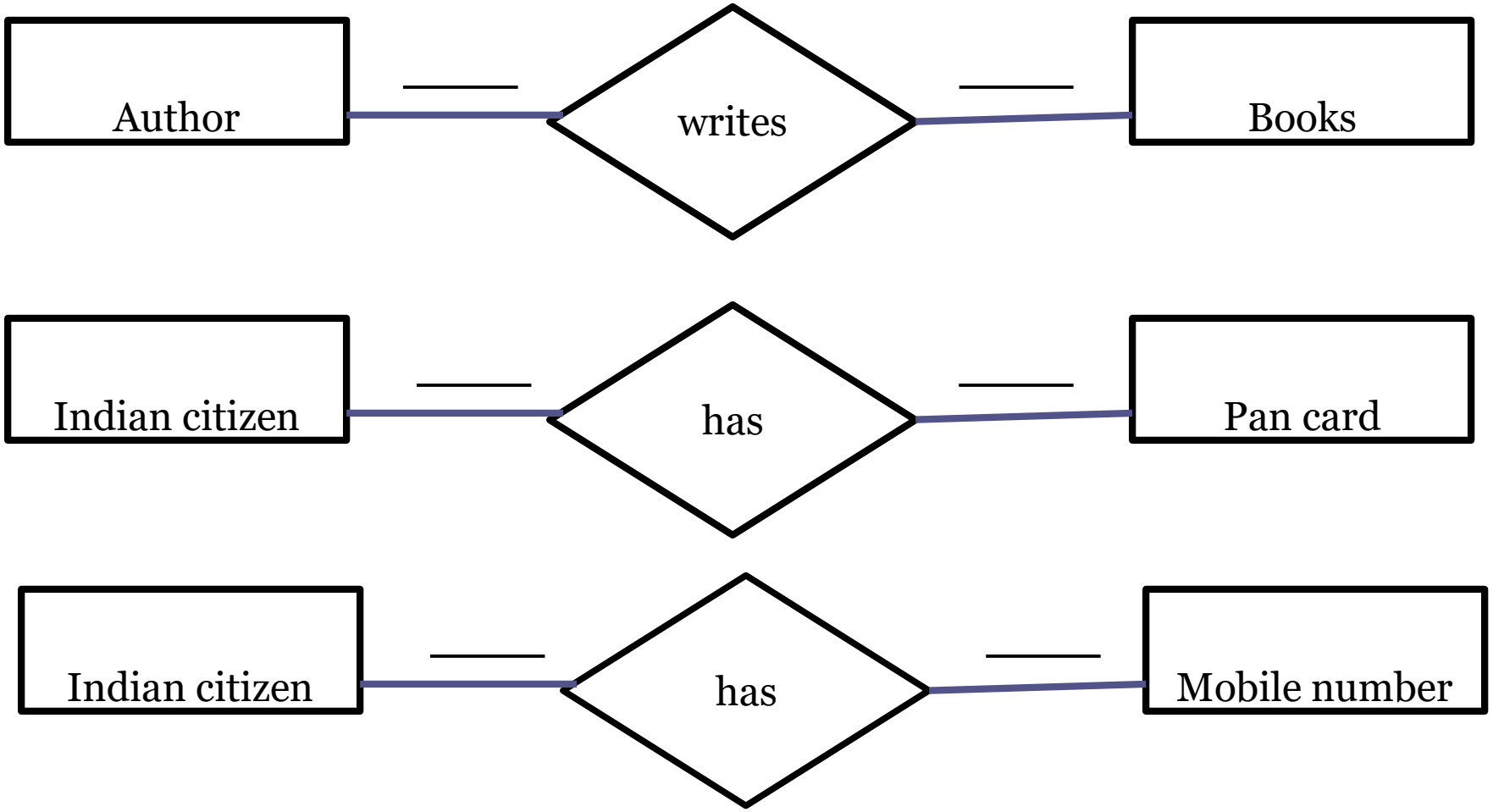
- It links the strong and weak entity and is represented by a double diamond sign.
- Let us see with an example to link both the entities using Identifying Relationships:



Above we saw that, Dependent Name could not exist on its own but in relationship to a Professor.

Professor	Strong Entity
Dependent	Weak Entity
Partial Key (Weak Entity)	Name
Primary Key (Strong Entity)	ID

Exercise

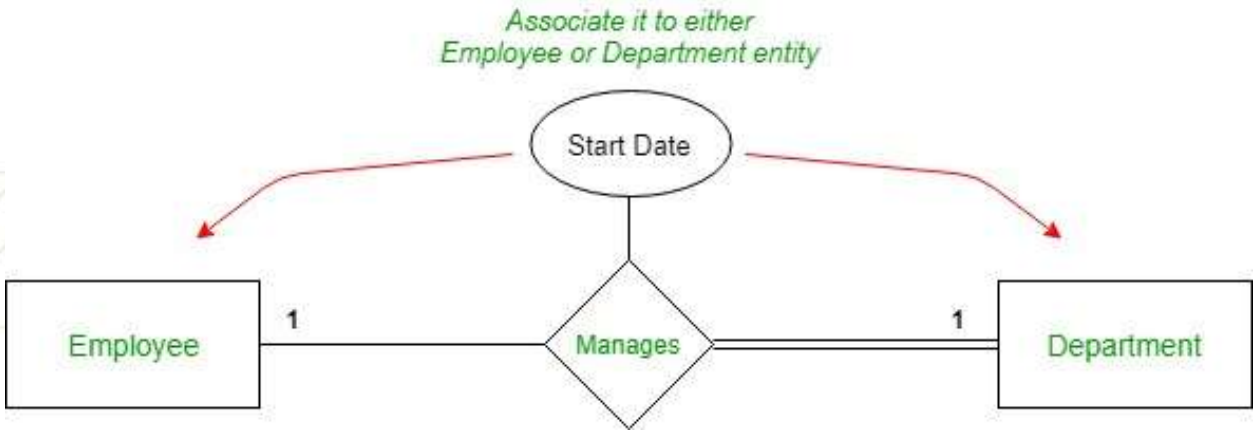

Author
writes
Books
Indian citizen
has
Pan card
Indian citizen
has
Mobile number

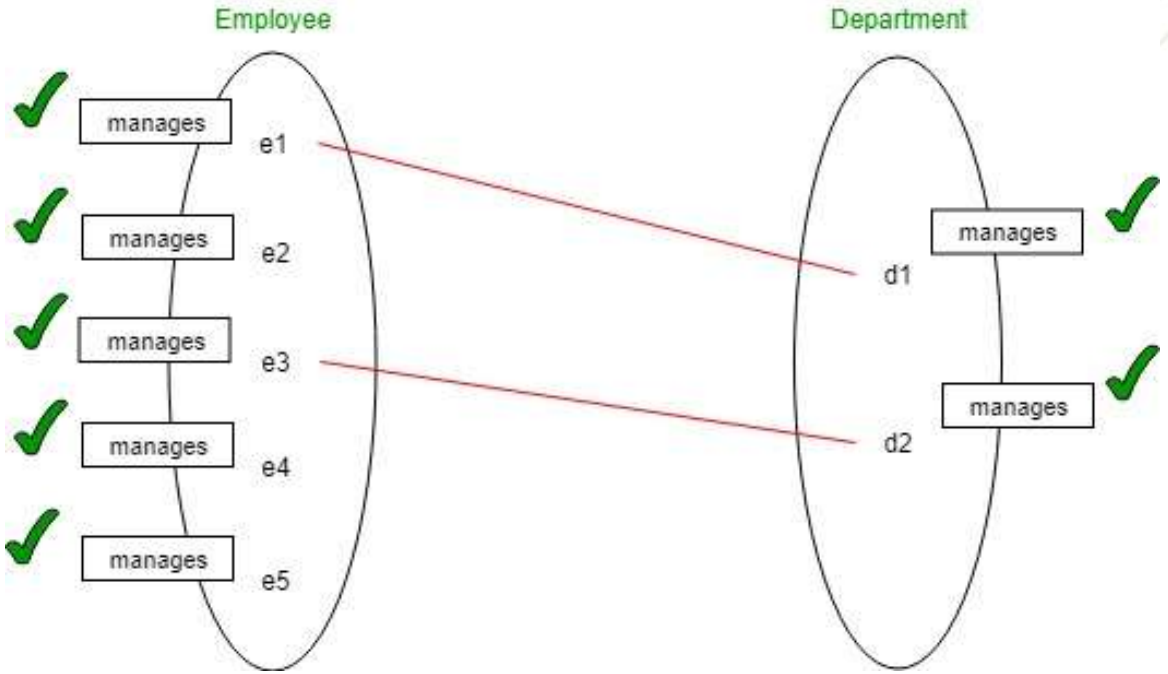
Attributes to Relationships in ER Model

In ER model, entities have attributes which can be of various types like single-valued, multi-valued, composite, simple, stored, derived and complex. **But relationships can also have attributes associated to them.** Generally it is not recommended to give attributes to the relationships if not required because while converting the ER model into Relational model, things may get complex and we may require to create a separate table for representing the relationship. Let us see various cases and when we need to give attributes to the relationship with the help of examples:

1. One to one relationship:

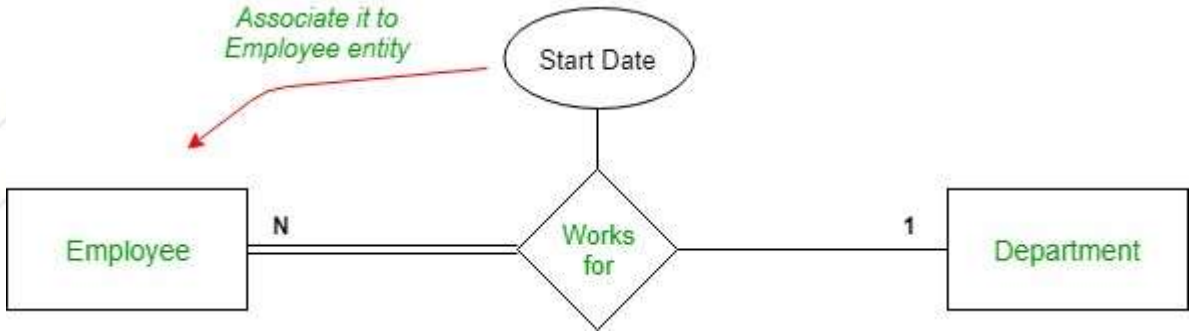
In an organisation an employee manages a department and each department is managed by some employee. So, there is a total participation of em>Department entity and there is *one to one* relationship between the given entities. Now, if we want to store the *Start_Date* from which the employee started managing the department then we may think that we can give the *Start_Date* attribute to the relationship *manages*. But, in this case we may avoid it by associating the *Start_Date* attribute to either *Employee* or *Department* entity.

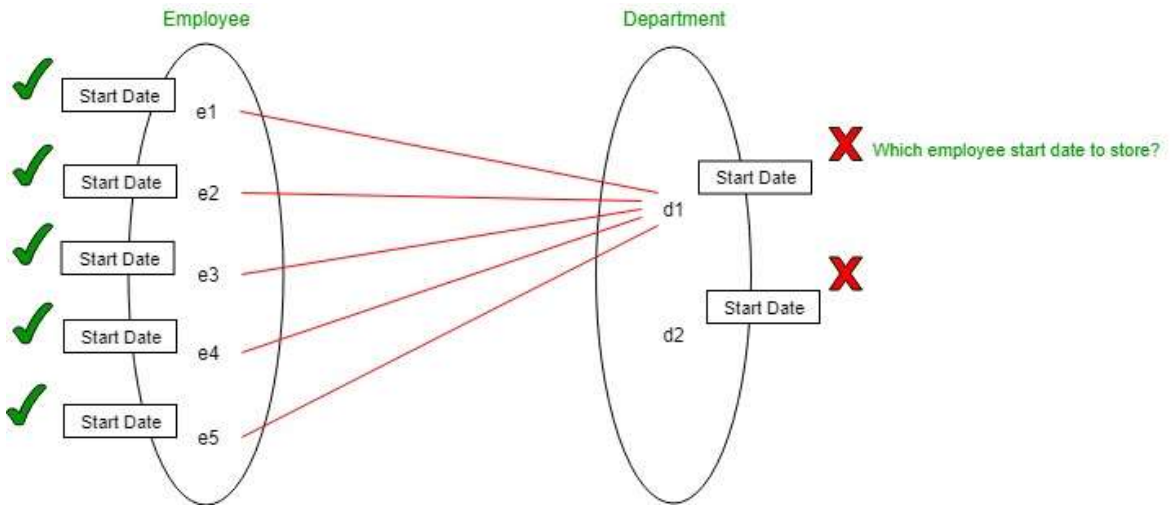




2. One to many relationship:

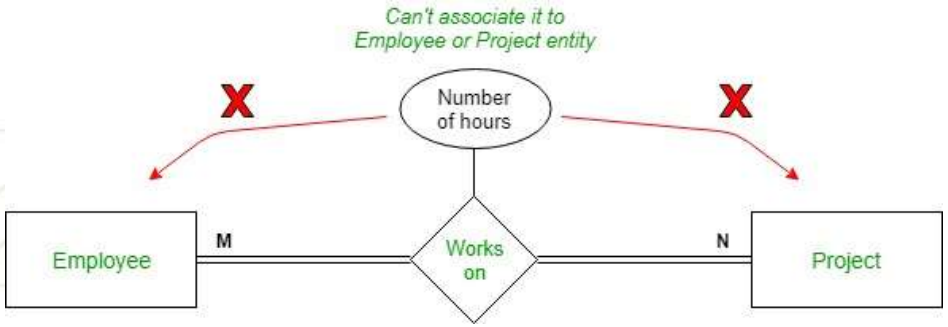
In an organisation many employees can work for a department but each employee can work for only a single department. So, there is a *one to many* relationship between the entities. Now if we want to store the *Start_Date* when employee started working for the department, **then instead of assigning it to the relationship we should assign it to the *Employee* entity**. Assigning it to the *employee* entity makes sense as each employee can work for only single department but on the other hand one department can have many employees working under it and hence, it wouldn't make sense if we assign Start_Date attribute to Department.

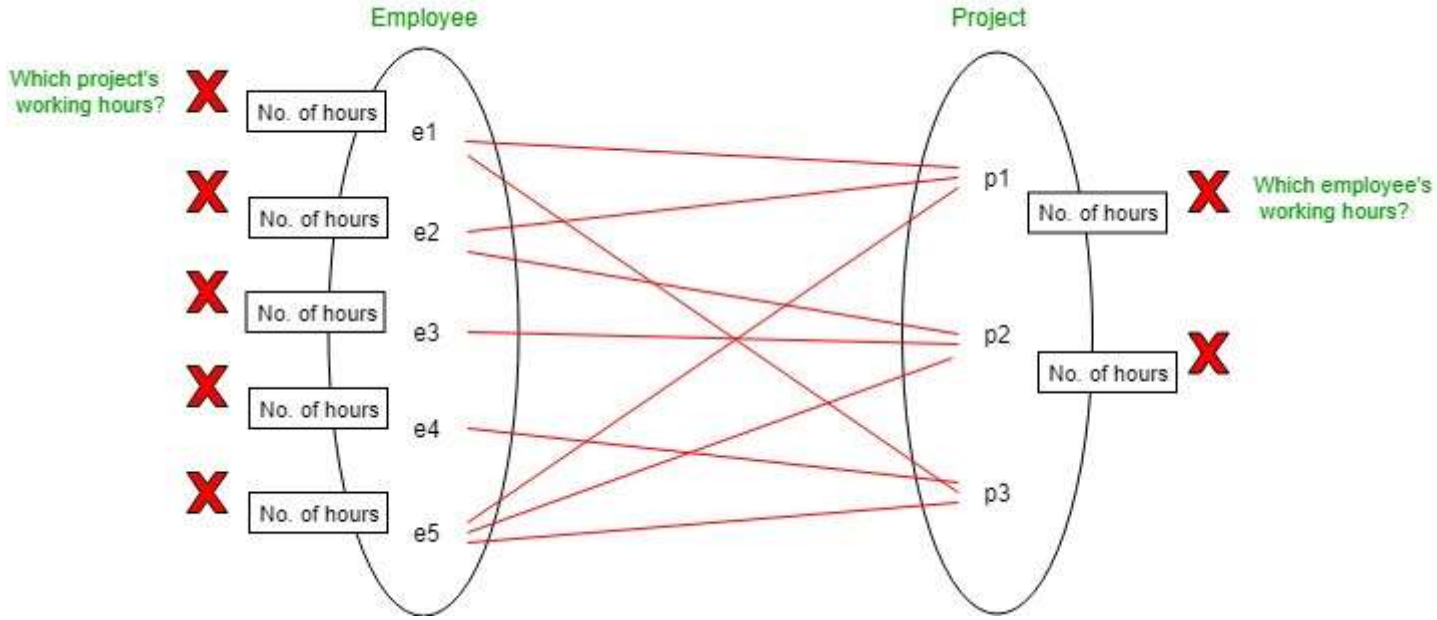




Many to many relationship:

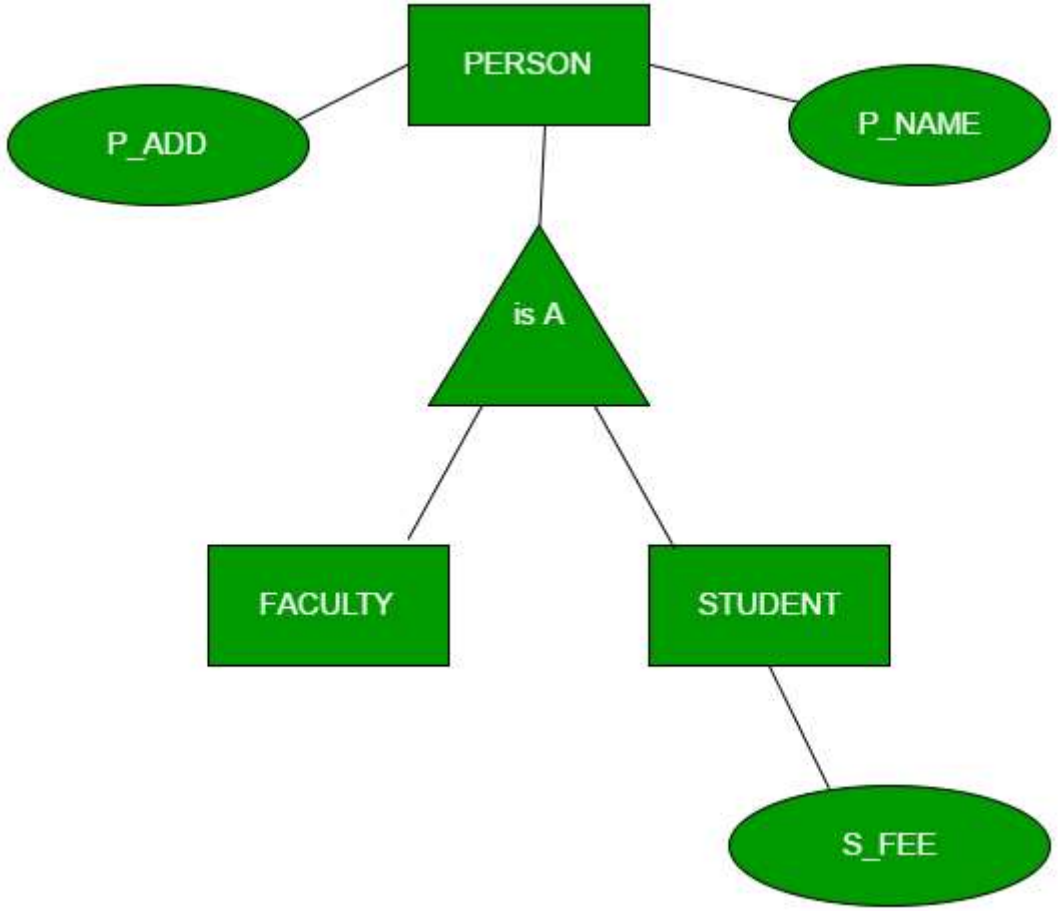
In an organisation **an employee can work on many projects** simultaneously and **each project can have many employees working on it.** Hence, it's a ***many to many*** **relationship.** So here assigning the *Number_of_Working_hours* to the employee will not work as the question will be that it will store which project's working hours because a single employee can work on multiple projects. Similar the case with the *project* entity. Hence, we are forced to assign the ***Number_of_Working_hours*** attribute to the relationship**.**





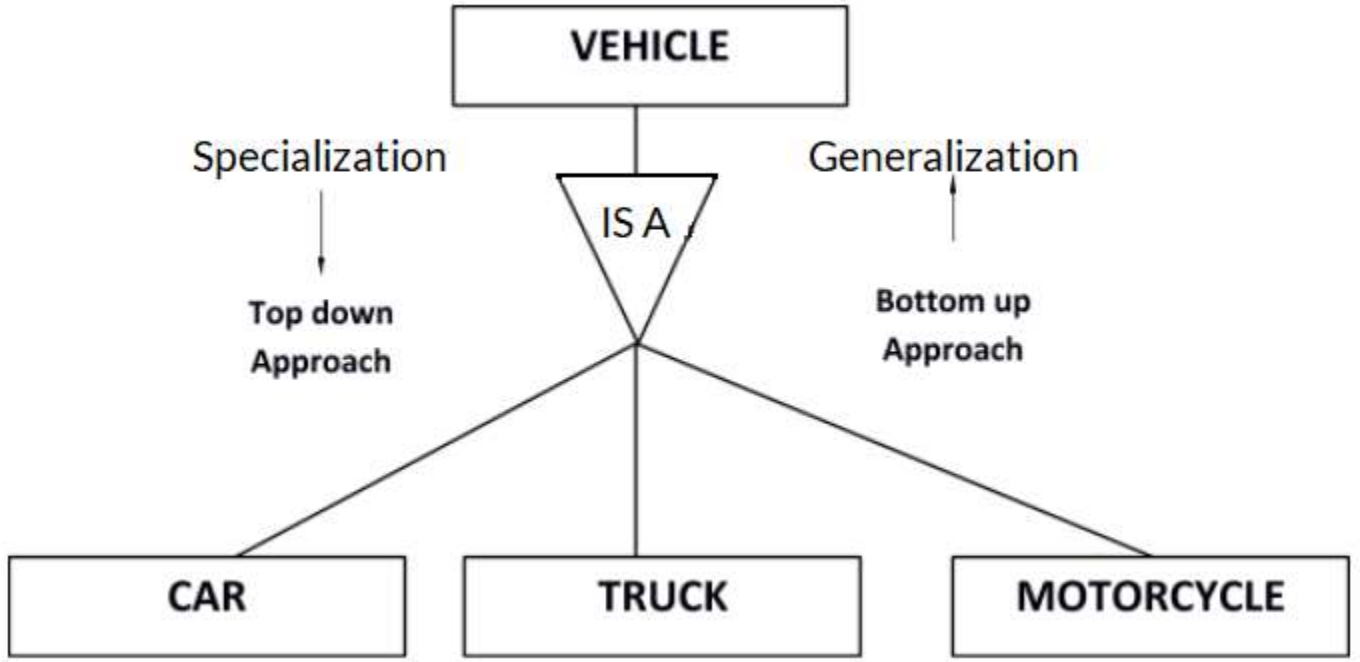
Generalization –

Generalization is the process of extracting common properties from a set of entities and create a generalized entity from it. It is a bottom-up approach in which two or more entities can be generalized to a higher level entity if they have some attributes in common. For Example, STUDENT and FACULTY can be generalized to a higher level entity called PERSON as shown in Figure 1. In this case, common attributes like P_NAME, P_ADD become part of higher entity (PERSON) and specialized attributes like S_FEE become part of specialized entity (STUDENT).



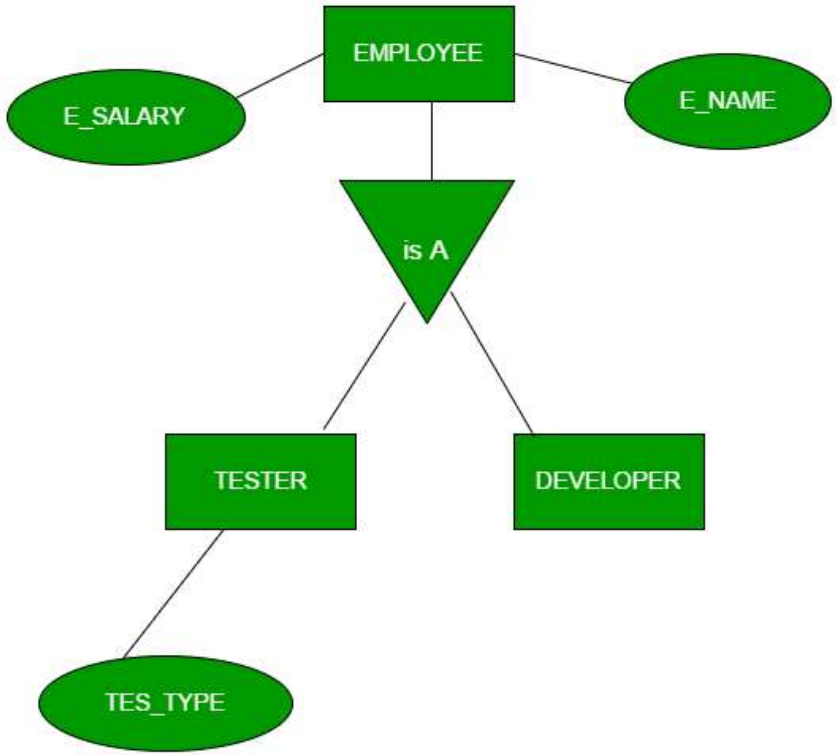
Generalization

- Generalization is a process of generalizing an entity which contains generalized attributes or properties of generalized entities. The entity that is created will contain the common features. Generalization is a Bottom up process.
- We can have three sub entities as Car, Truck, Motorcycle and these three entities can be generalized into one general super class as Vehicle.



Specialization

In specialization, an entity is divided into sub-entities based on their characteristics. It is a top-down approach where higher level entity is specialized into two or more lower level entities. For Example, EMPLOYEE entity in an Employee management system can be specialized into DEVELOPER, TESTER etc. as shown in Figure 2. In this case, common attributes like E_NAME, E_SAL etc. become part of higher entity (EMPLOYEE) and specialized attributes like TES_TYPE become part of specialized entity (TESTER).



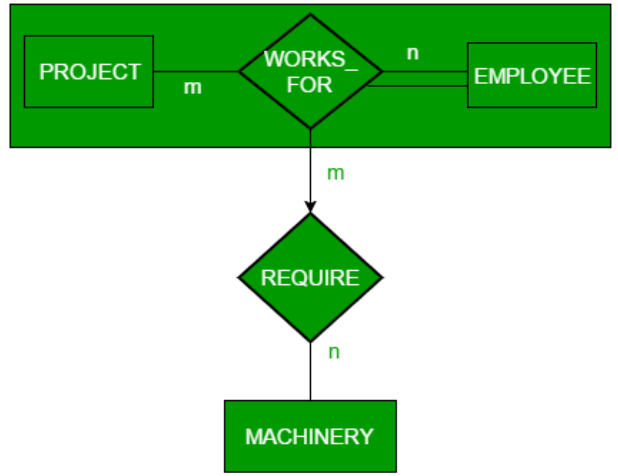
Specialization

Aggregation

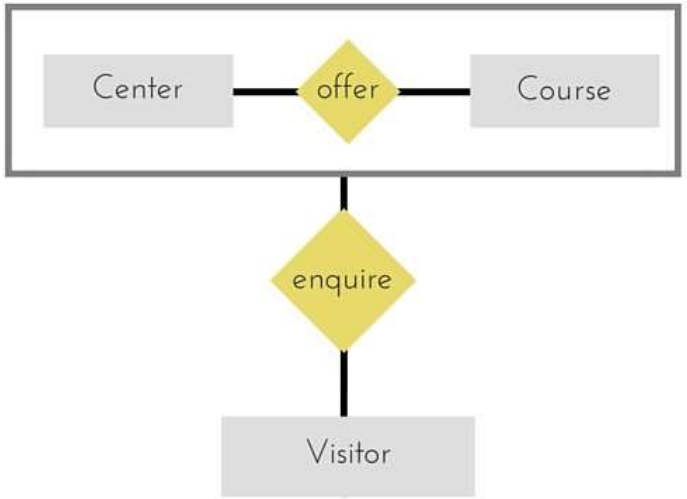
Aggregation is a process when relation between two entities is treated as a **single entity**.

An ER diagram is not capable of representing relationship between an entity and a relationship which may be required in some scenarios. **In those cases, a relationship with its corresponding entities is aggregated into a higher level entity.** Aggregation is an abstraction through which we can represent relationships as higher level entity sets.

For Example, Employee working for a project may require some machinery. So, REQUIRE relationship is needed between relationship WORKS_FOR and entity MACHINERY. Using aggregation, WORKS_FOR relationship with its entities EMPLOYEE and PROJECT is aggregated into single entity and relationship REQUIRE is created between aggregated entity and MACHINERY.



Aggregation



Question: Construct an E-R diagram

VIPS Technical Campus
योगः कर्मसु कौशलम्
SCHOOL OF ENGINEERING AND TECHNOLOGY

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

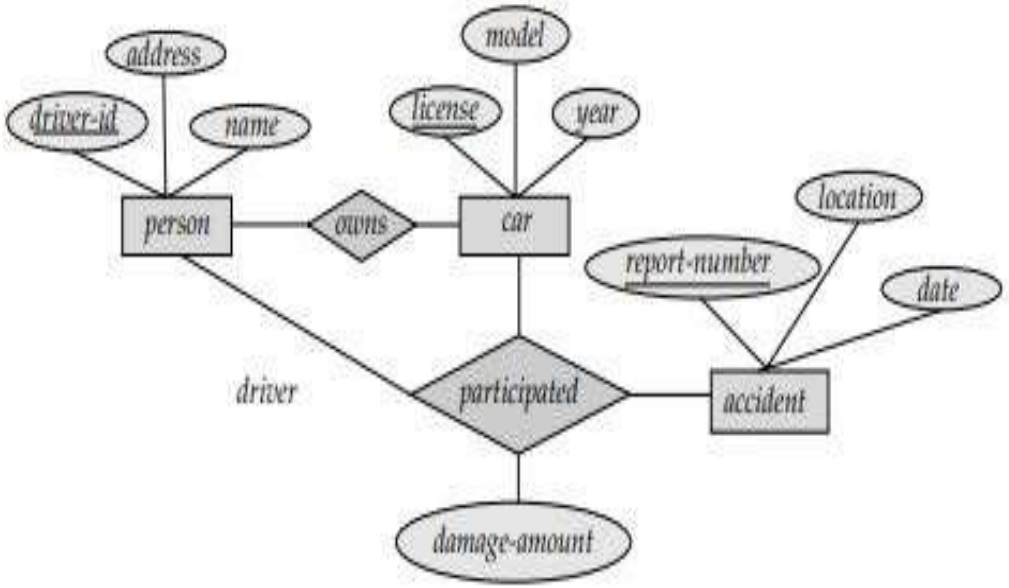


Figure 2.1 E-R diagram for a Car-insurance company.

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

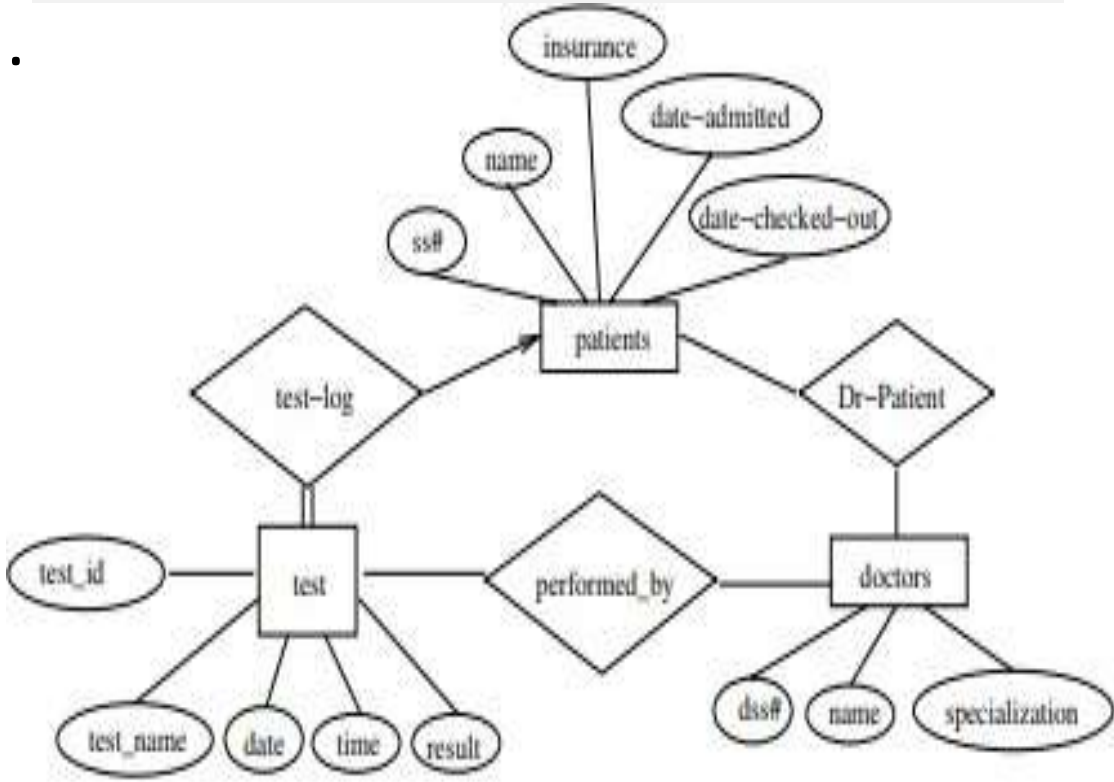


Figure 2.2 E-R diagram for a hospital.

Question: ER Model

A university registrar's office maintains data about the following entities: (a) courses, including number, title, credits, syllabus, and prerequisites; (b) course offerings, including course number, year, semester, section number, instructor(s), timings, and classroom; (c) students, including student-id, name, and program; and (d) instructors, including identification number, name, department, and title. Further, the enrollment of students in courses and grades awarded to students in each course they are enrolled for must be appropriately modeled.

Construct an E-R diagram forthe registrar's office. Document all assumptions that you make about the mapping constraints.

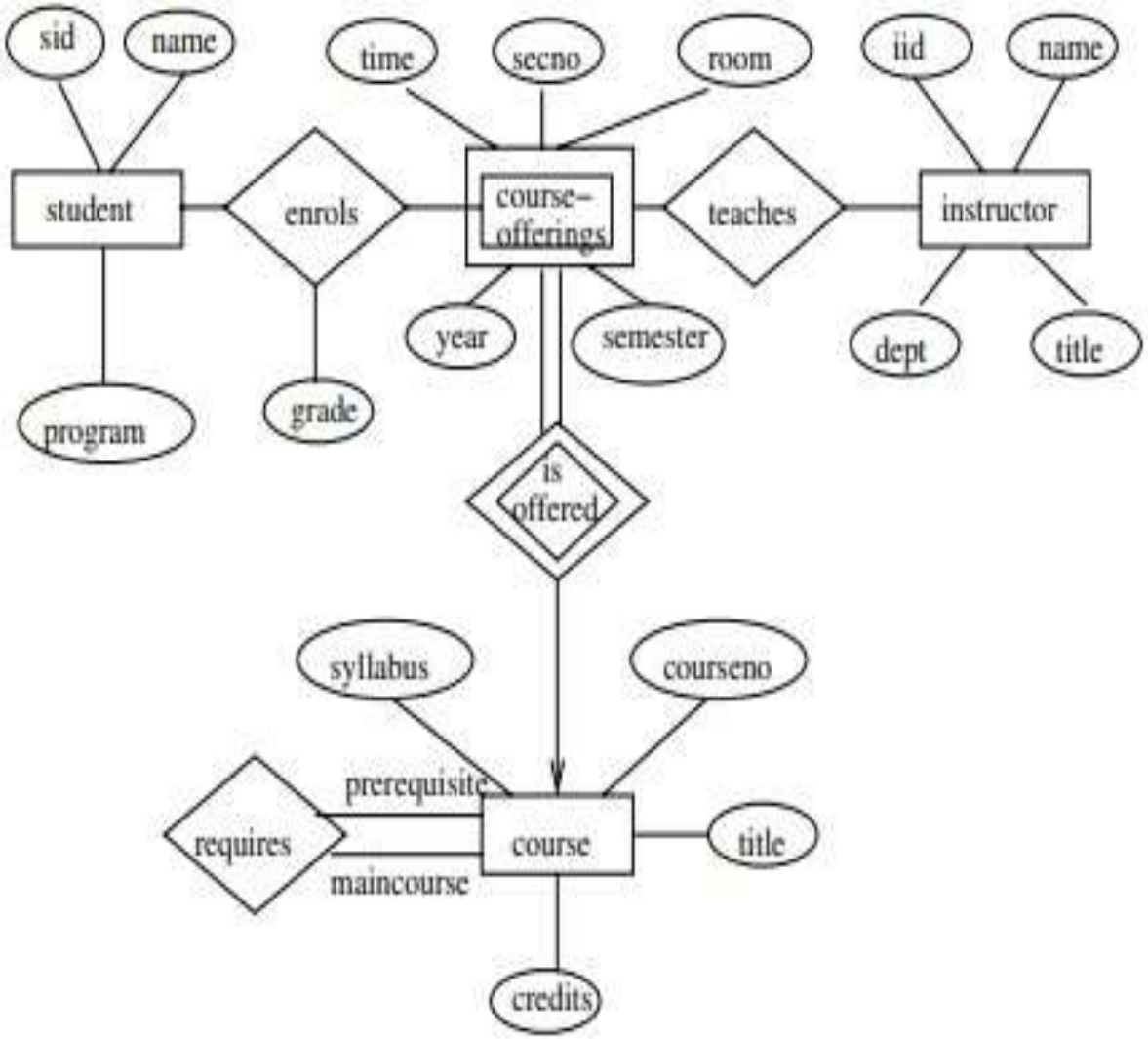


Figure 2.3 E-R diagram for a university.

- Consider a database used to record the marks that students get in different exams of different course offerings.
- Construct an E-R diagram that models exams as entities, and uses a ternary relationship, for the above database

.

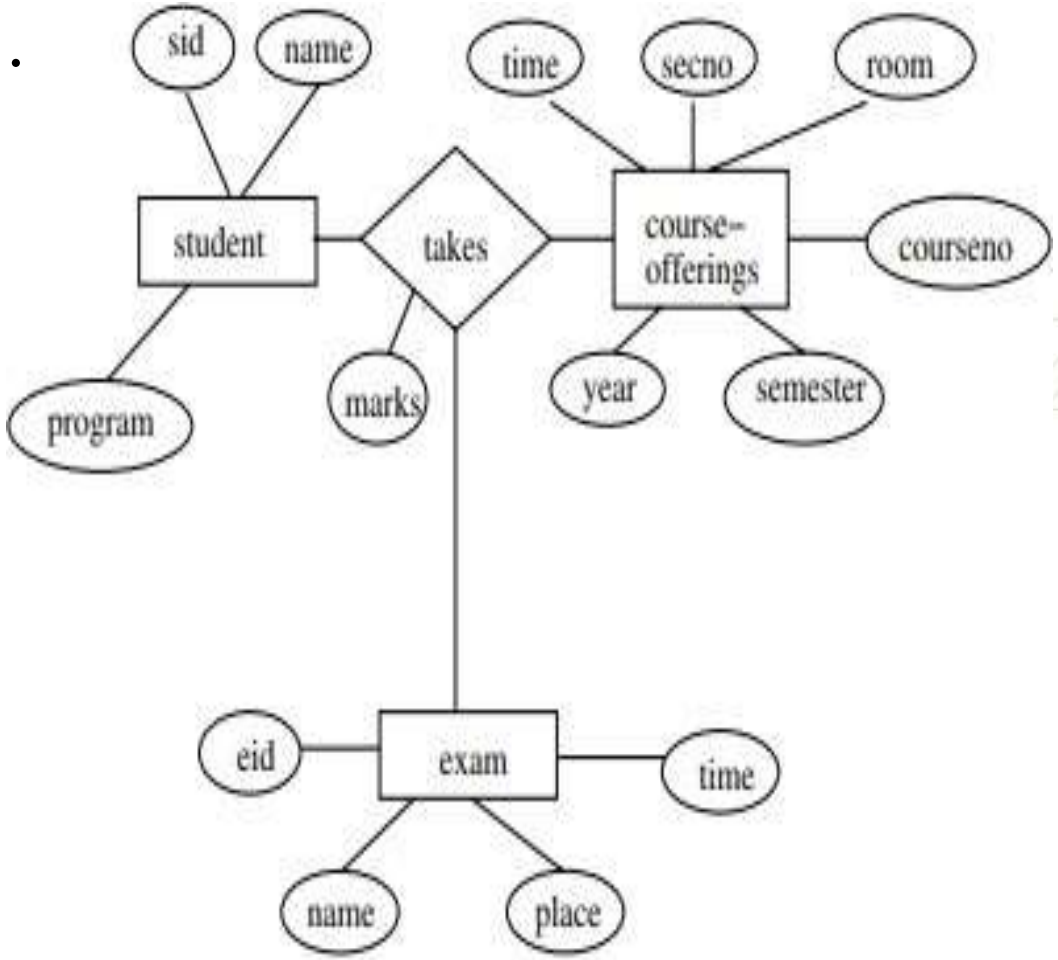


Figure 2.4 E-R diagram for marks database.

Converting ER Diagrams to Tables-

VIPS Technical Campus — SCHOOL OF ENGINEERING AND TECHNOLOGY

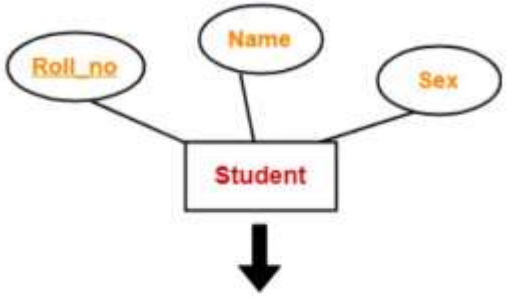
After designing an ER Diagram,

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

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

A strong entity set with only simple attributes will require only one table in relational model.

- Attributes of the table will be the attributes of the entity set.
- The primary key of the table will be the key attribute of the entity set.



Roll_no	Name	Sex

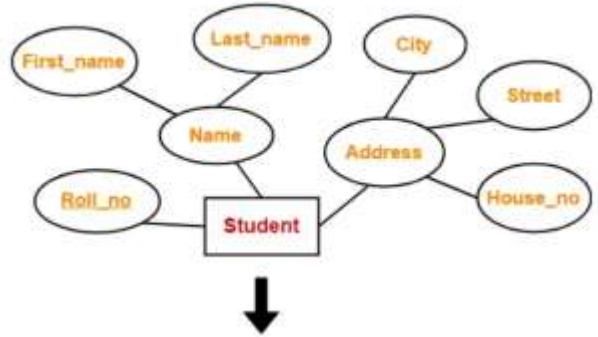
Schema : Student (Roll_no , Name , Sex)

https://www.gatevidyalay.com/er-diagrams-to-tables/

Rule-02: For Strong Entity Set With Composite Attributes-

A strong entity set with any number of composite attributes will require only one table in relational model.

- While conversion, simple attributes of the composite attributes are taken into account and not the composite attribute itself.



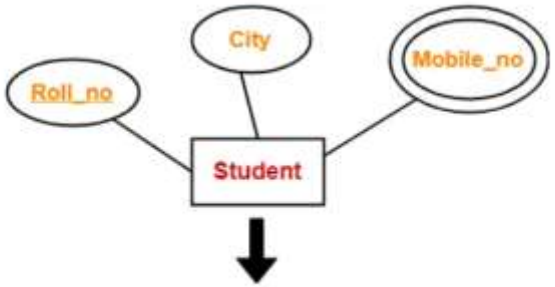
Roll_no	First_name	Last_name	House_no	Street	City

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

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

•One table will contain all the simple attributes with the primary key.

•Other table will contain the primary key and all the multi valued attributes.



Roll_no	City

Roll_no	Mobile_no

NOTE:
If we consider the overall ER diagram, three tables will be required in relational model-
•One table for the entity set "Employee"
•One table for the entity set "Department"
•One table for the relationship set "Works in"

Rule-04: Translating Relationship Set into a Table

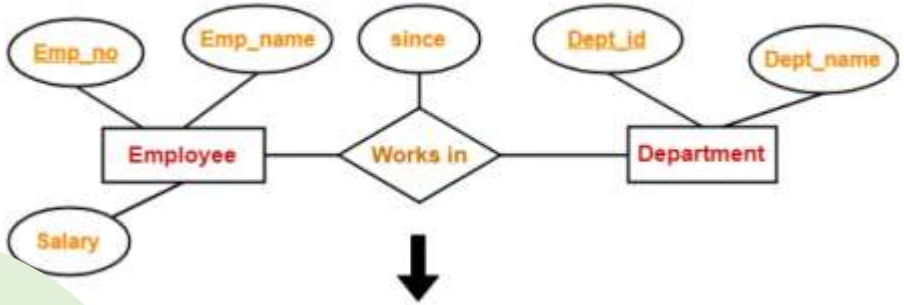
A relationship set will require one table in the relational model.

Attributes of the table are-

•Primary key attributes of the participating entity sets

•Its own descriptive attributes if any.

Set of non-descriptive attributes will be the primary key.

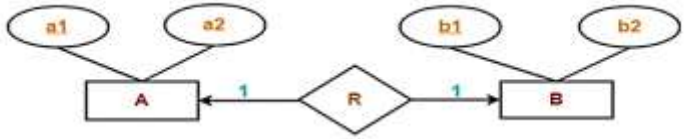


Emp_no	Dept_id	since

Schema : Works in (Emp_no , Dept_id , since)

Rule-05: For Binary Relationships With Cardinality Ratios-

Case-04: Binary relationship with cardinality ratio 1:1



Here, two tables will be required. Either combine 'R' with 'A' or 'B'

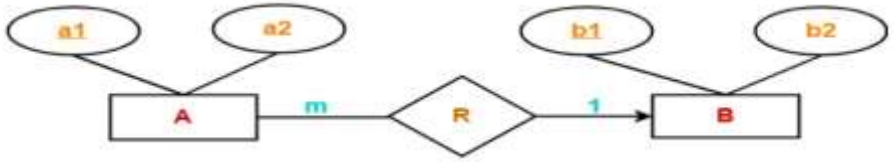
Way-01:

1. AR (a1 , a2 , b1)
2. B (b1 , b2)

Way-02:

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

Case-03: For Binary Relationship With Cardinality Ratio m:1

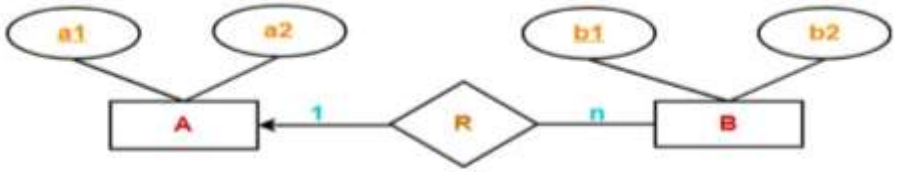


Here, two tables will be required-

1. AR (a1 , a2 , b1)
2. B (b1 , b2)

NOTE- Here, combined table will be drawn for the entity set A and relationship set R.

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

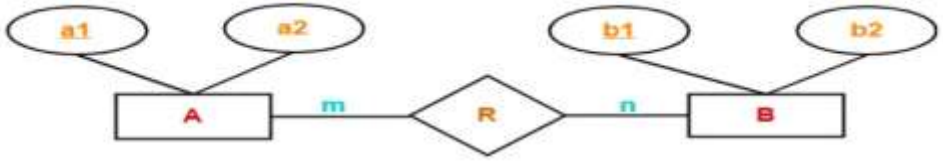


Here, two tables will be required-

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

NOTE- Here, combined table will be drawn for the entity set B and relationship set R.

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



Here, three tables will be required-

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

Thumb Rules to Remember

While determining the minimum number of tables required for binary relationships with given cardinality ratios, following thumb rules must be kept in mind-

•For binary relationship with cardinality ration m : n , separate and individual tables will be drawn for each entity set and relationship.

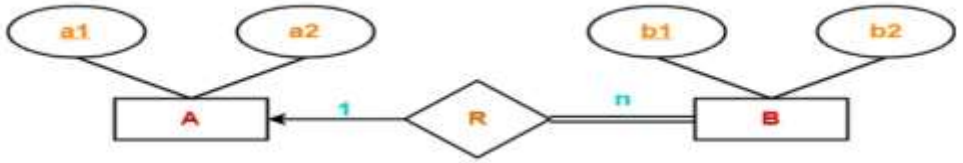
•For binary relationship with cardinality ratio either m : 1 or 1 : n , always remember “many side will consume the relationship” i.e. a combined table will be drawn for many side entity set and relationship set.

•For binary relationship with cardinality ratio 1 : 1 , two tables will be required. You can combine the relationship set with any one of the entity sets.

Rule-06: For Binary Relationship With Both Cardinality Constraints and Participation Constraints-

•Cardinality constraints will be implemented as discussed in Rule-05.
•Because of the total participation constraint, foreign key acquires **NOT NULL** constraint i.e. now foreign key can not be null.

Case-01: For Binary Relationship With Cardinality Constraint and Total Participation Constraint From One Side-



Because cardinality ratio = 1 : n , so we will combine the entity set B and relationship set R.

Then, two tables will be required-

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

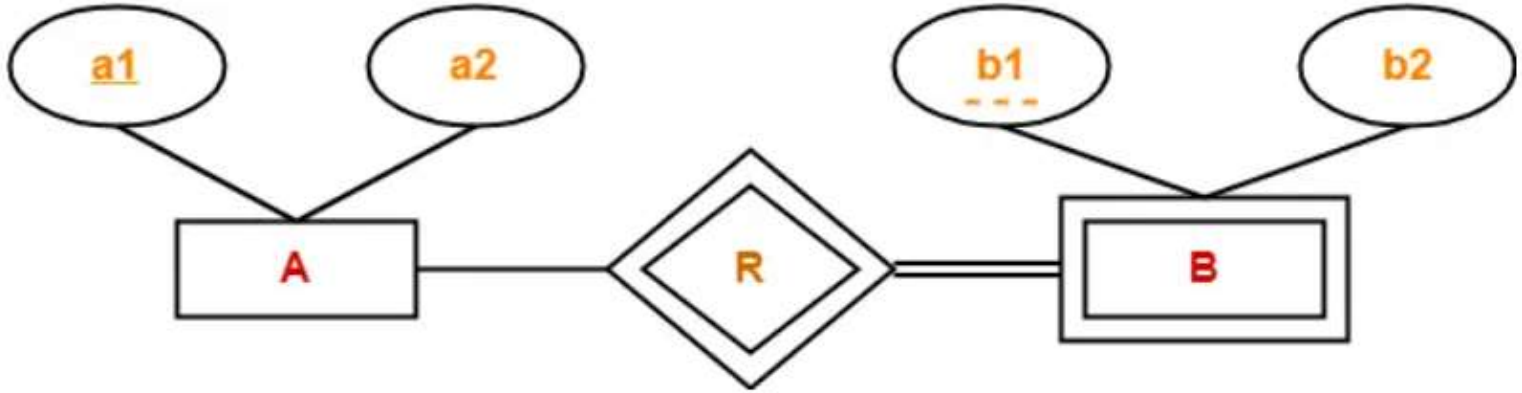
Because of total participation, foreign key a1 has acquired NOT NULL constraint, so it can't be null now.

Case-02: For Binary Relationship With Cardinality Constraint and Total Participation Constraint From Both Sides, One side-

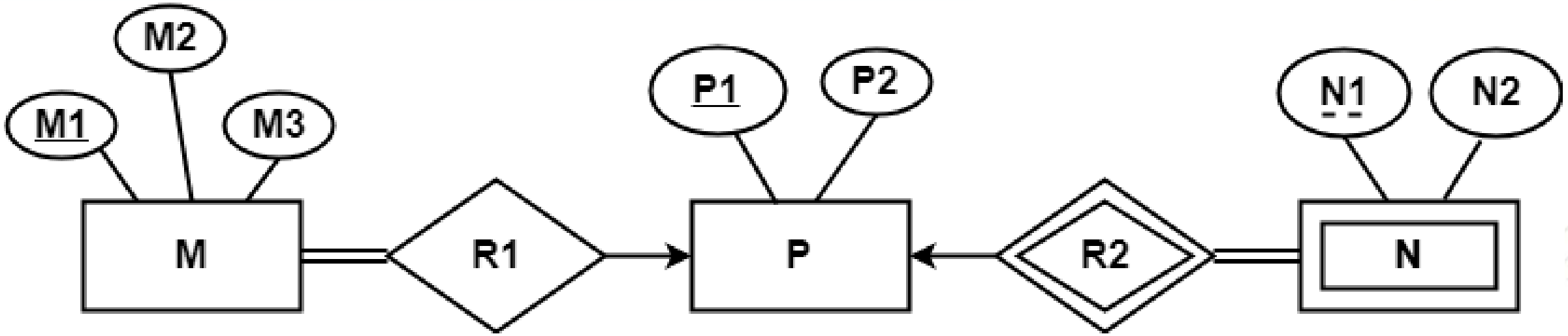
Here, Only one table is required.
•ARB (a1 , a2 , b1 , b2)

Rule-07: For Binary Relationship With Weak Entity Set-

Weak entity set always appears in association with identifying relationship with total participation constraint.



Here, two tables will be required-
1.A (a1 , a2)
2.BR (a1 , b1 , b2)

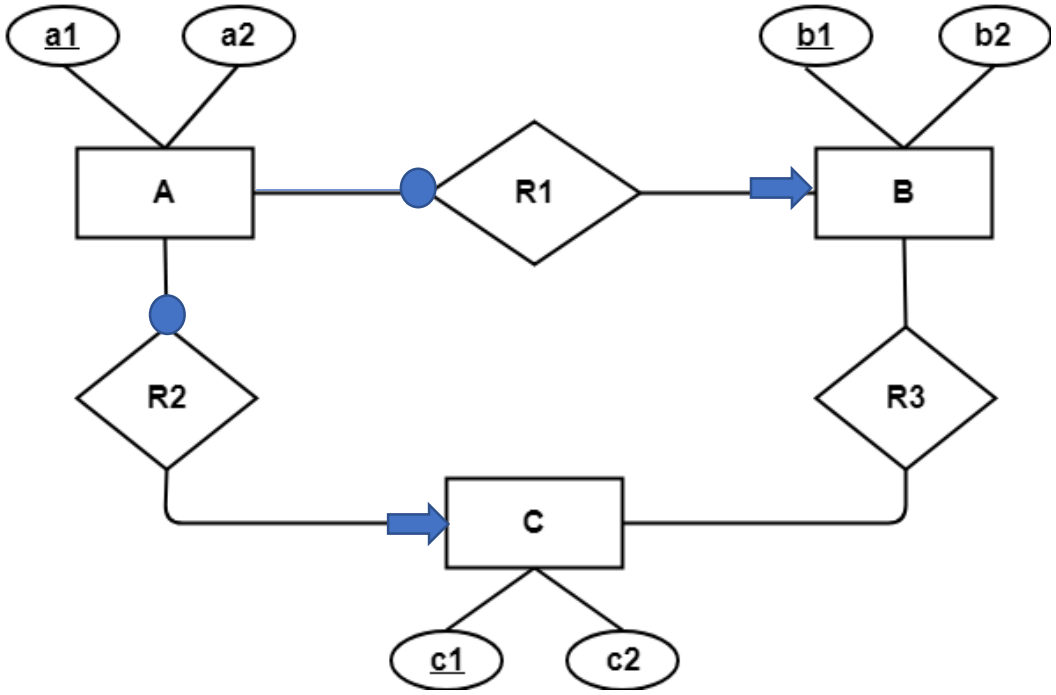


Applying the rules, minimum 3 tables will be required-

Applying the rules, minimum 3 tables will be required-

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

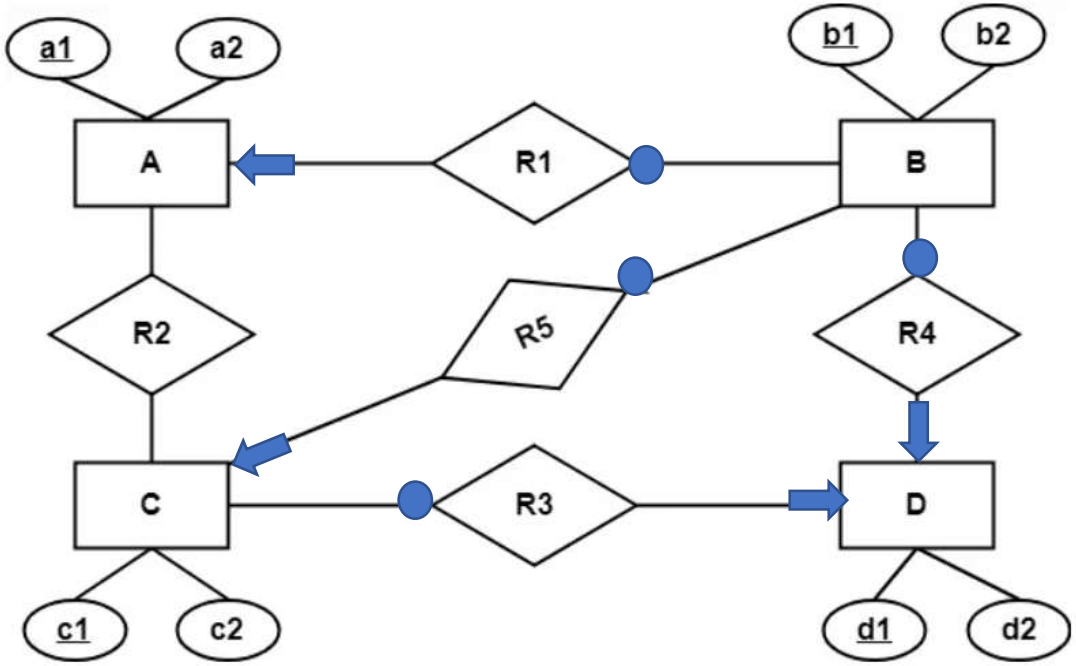
Find the minimum number of tables required to represent the given ER diagram in relational model-



Solution-

Applying the rules, minimum 4 tables will be required-

- AR1R2 (a1 , a2 , b1 , c1)
- B (b1 , b2)
- C (c1 , c2)
- R3 (b1 , c1)



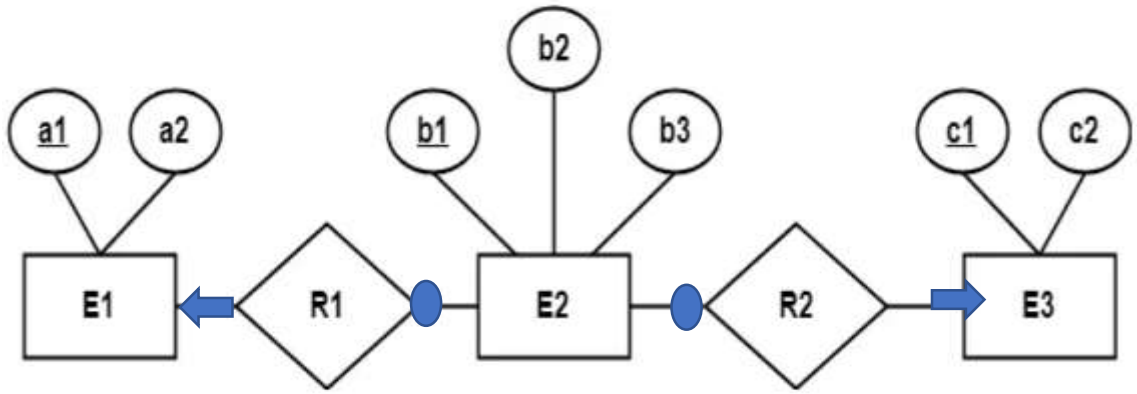
Solution-

Applying the rules, minimum 5 tables will be required-

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

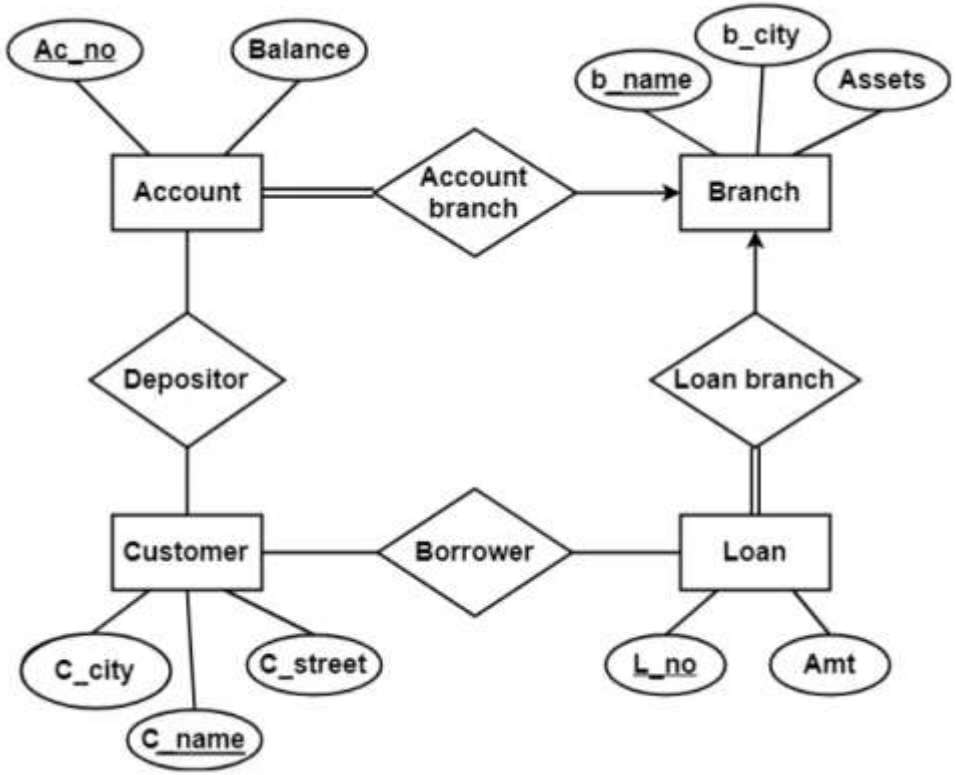
Problem-04 & 05:

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



Applying the rules, minimum 3 tables will be required-

- E1 (a1 , a2)
- E2R1R2 (b1 , b2 , a1 , c1 , b3)
- E3 (c1 , c2)



Solution-

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

- Account (Ac_no , Balance , b_name)
- Branch (b_name , b_city , Assets)
- Loan (L_no , Amt , b_name)
- Borrower (C_name , L_no)
- Customer (C_name , C_street , C_city)
- Depositor (C_name , Ac_no)

Problem-

Suppose you are given the following requirements for a simple database for the National Hockey League (NHL):

· the NHL has many teams,

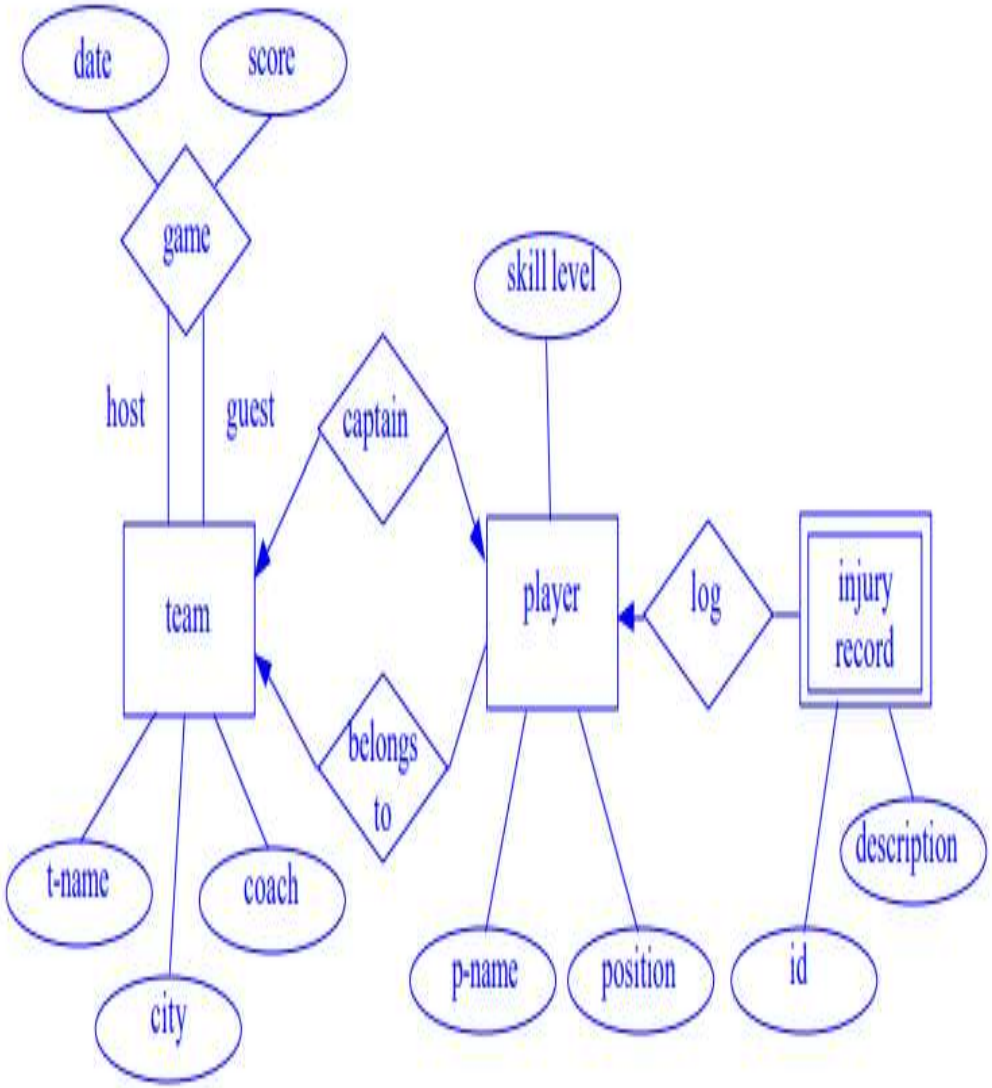
· each team has a name, a city, a coach, a captain, and a set of players,

· each player belongs to only one team,

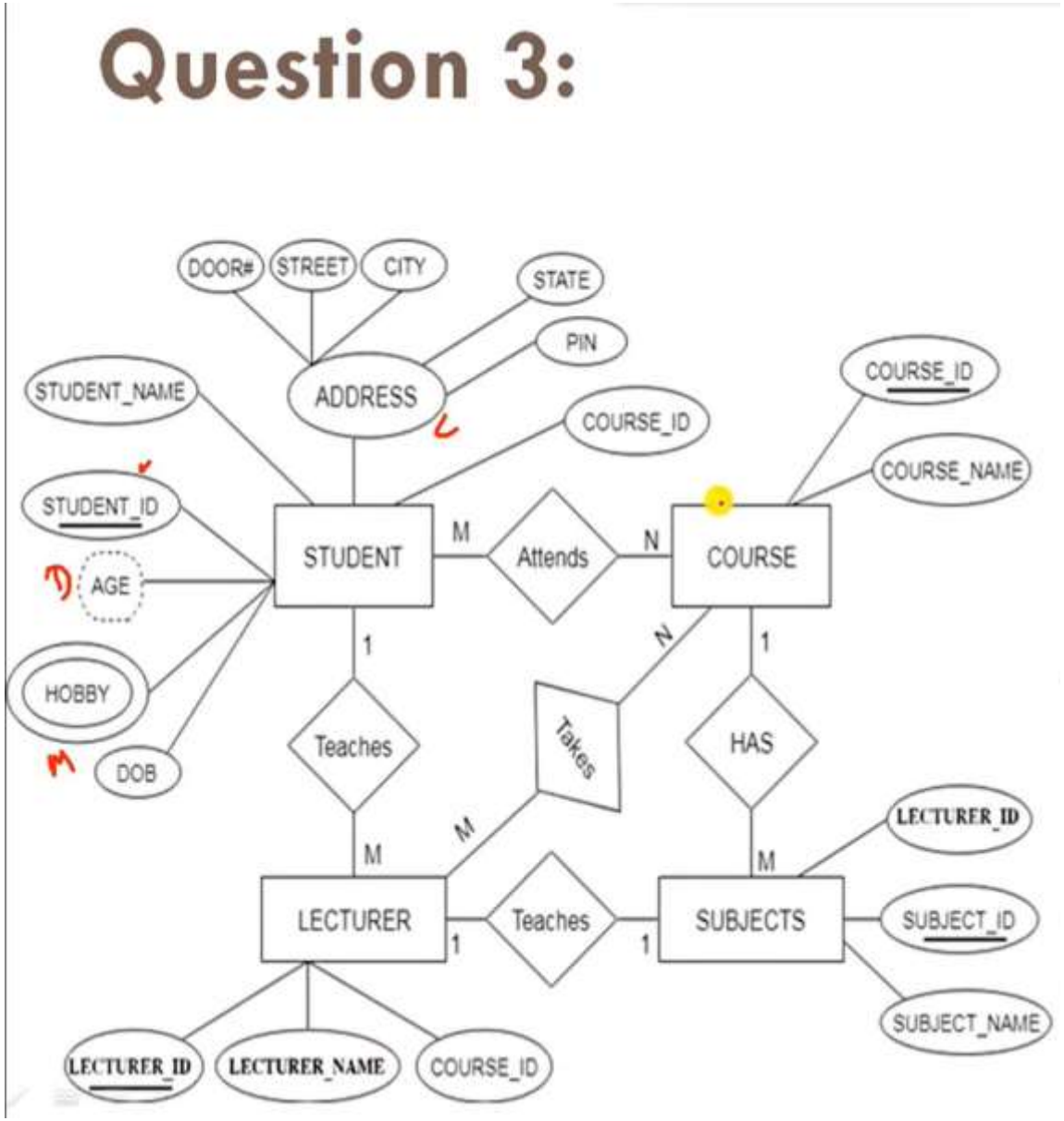
· each player has a name, a position (such as left wing or goalie), a skill level, and a set of injury records,

· a team captain is also a player,

· a game is played between two teams (referred to as host_team and guest_team) and has a date (such as May 11th, 1999) and a score (such as 4 to 2).



Problem-



Minimum 7 tables will be required:

1. **Student** (Student_Id, Student_Name, DOB, Door#, Street, City, State, Pin, Course_Id)
2. **Lecturer** (Lecturer_Id, Lecturer_Name, Course_Id, Student_Id, Subject_Id)
3. **Course** (Course_Id, Course_Name)
4. **Subjects** (Subject_Id, Subject_Name, Lecturer_Id, Course_Id)
5. **Hobby** (Subject_Id, Hobby)
6. **Attends** (Subject_Id, Course_Id)
7. **Takes** (Lecturer_Id, Course_Id)

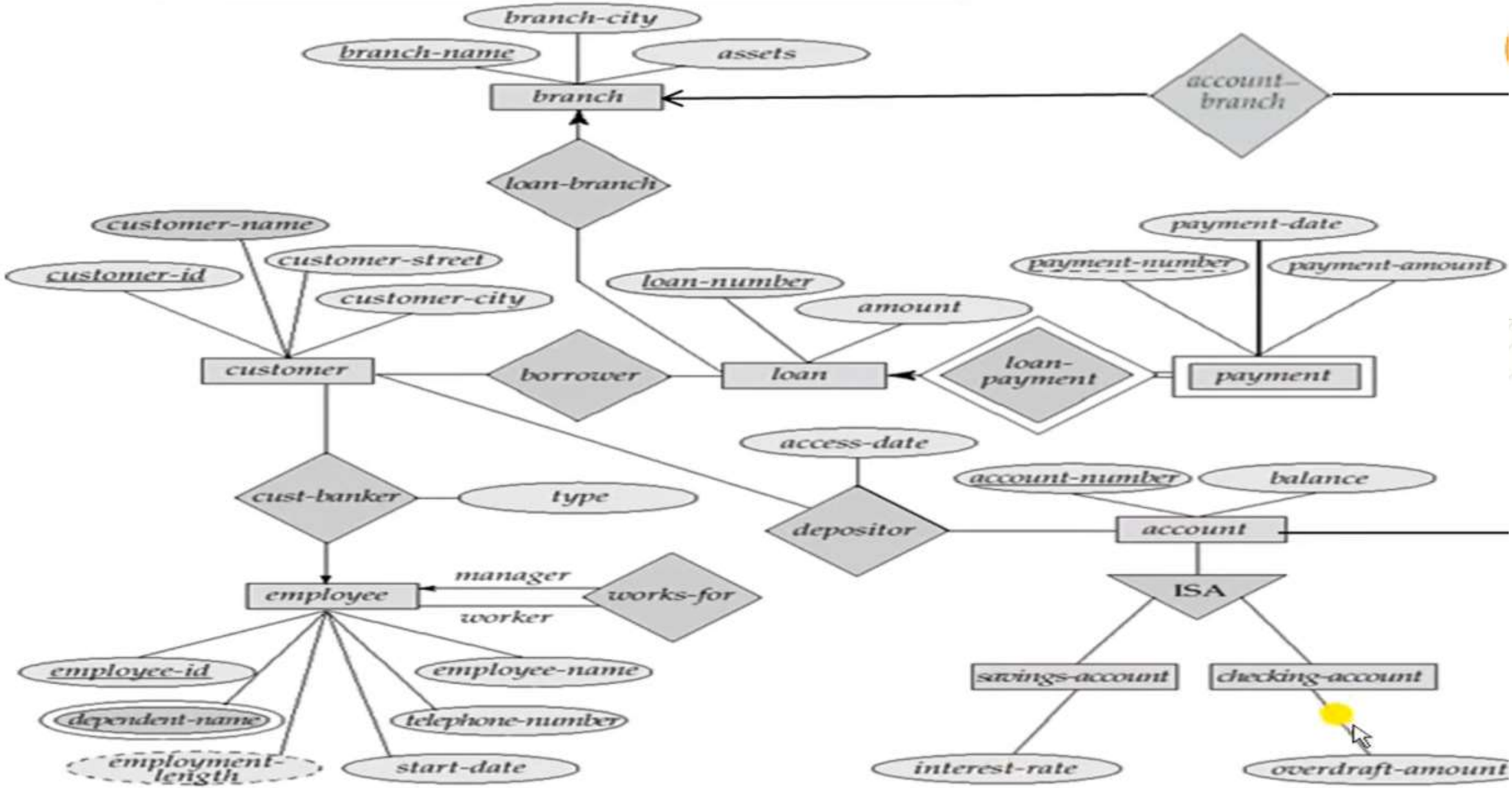
e bank is organized into **branches**. Each branch is located in a particular ***city*** and is identified by a que ***name***. The bank monitors the ***assets*** of each branch.

nk **customers** are identified by their ***customer id*** values. The bank stores each customer's ***name*** and the ***eet*** and ***city*** where the customer lives. Customers may have **accounts** and can take out **loans**. A customer y be associated with a particular bank **employee**, who may act as a loan officer or personal banker for t customer.

nk **employees** are identified by their ***employee id*** values. The bank administration stores the ***name*** and ***ephone number*** of each employee, the names of the employee's ***dependents***, and the employee id number the employee's manager. The bank also keeps the track of the employee's ***start date*** and, thus, ***length of ployment***.

e bank offers two types of accounts - savings and checking accounts. **Accounts** can be held by more than e **customer**, and a customer can have more than one account. Each account is assigned a unique ***account nber***. The bank maintains a record of account's ***balance*** and the ***most recent date*** on which the account s accessed by each customer holding the account. In addition, each savings account has an interest rate d overdrafts are recorded for each checking account.

oan originates at a particular **branch** and can be held by one or more **customers**. A loan is identified by nique ***loan number***. For each loan, the bank keeps track of the loan ***amount*** and the **loan payments**. hough a loan payment number does not uniquely identify a particular payment among those for all the nk's loans, a payment number does identify a particular payment for a specific loan. The ***date*** and ***ount*** are recorded for each payment.


branch-city
branch-name
assets
branch
account-branch
loan-branch
customer-name
customer-id
customer-street
customer-city
customer
loan-number
amount
borrower
loan
loan-payment
payment-date
payment-number
payment-amount
payment
access-date
account-number
balance
cust-banker
type
depositor
account
manager
employee
works-for
worker
ISA
employee-id
employee-name
dependent-name
telephone-number
employment-length
start-date
savings-account
checking-account
interest-rate
overdraft-amount

VIPS Technical Campus
योग: कर्मसु कौशलम्
IN PURSUIT OF PERFECTION

SCHOOL OF ENGINEERING AND TECHNOLOGY

THANK YOU