DBMS

Department of Computer Science & Engineering

B. Tech (CSE)

UNIT-1

1.Overview of Database Management System:

Database:

The database is a collection of inter-related data which is used to retrieve, insert and delete the data efficiently. It is also used to organize the data in the form of a table, schema, views, and reports, etc.

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.

Database Management System:

Database management system is a software which is used to manage the database. For example: **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.

Characteristics of DBMS:

- It uses a digital repository established on a server to store and manage the information.
- It can provide a clear and logical view of the process that manipulates data.
- DBMS contains automatic backup and recovery procedures.
- It contains ACID properties which maintain data in a healthy state in case of failure.
- It can reduce the complex relationship between data.
- It is used to support manipulation and processing of data.
- It is used to provide security of data.
- It can view the database from different viewpoints according to the requirements of the user.

Advantages of DBMS OVER FILE PROCESSING SYSTEM:

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.

2. DBMS - Architecture:

The design of a DBMS depends on its architecture. It can be centralized or decentralized or hierarchical. The architecture of a DBMS can be seen as either single tier or multi-tier. An n-tier architecture divides the whole system into related but independent n modules, which can be independently modified, altered, changed, or replaced.

1-Tier architecture:

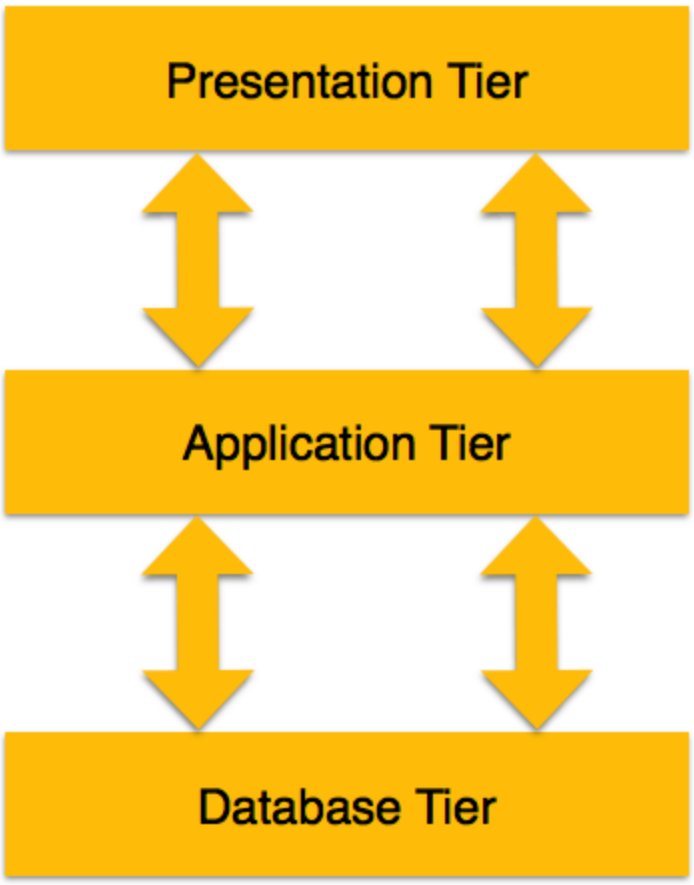
In 1-tier architecture, the DBMS is the only entity where the user directly sits on the DBMS and uses it. Any changes done here will directly be done on the DBMS itself. It does not provide handy tools for end-_users. Database designers and programmers normally prefer to use single-tier architecture.

2-Tier architecture:

If the architecture of DBMS is 2-tier, then it must have an application through which the DBMS can be accessed. Programmers use 2-tier architecture where they access the DBMS by means of an application. Here the application tier is entirely independent of the database in terms of operation, design, and programming.

3-Tier architecture:

A 3-tier architecture separates its tiers from each other based on the complexity of the users and how they use the data present in the database. It is the most widely used architecture to design a DBMS.



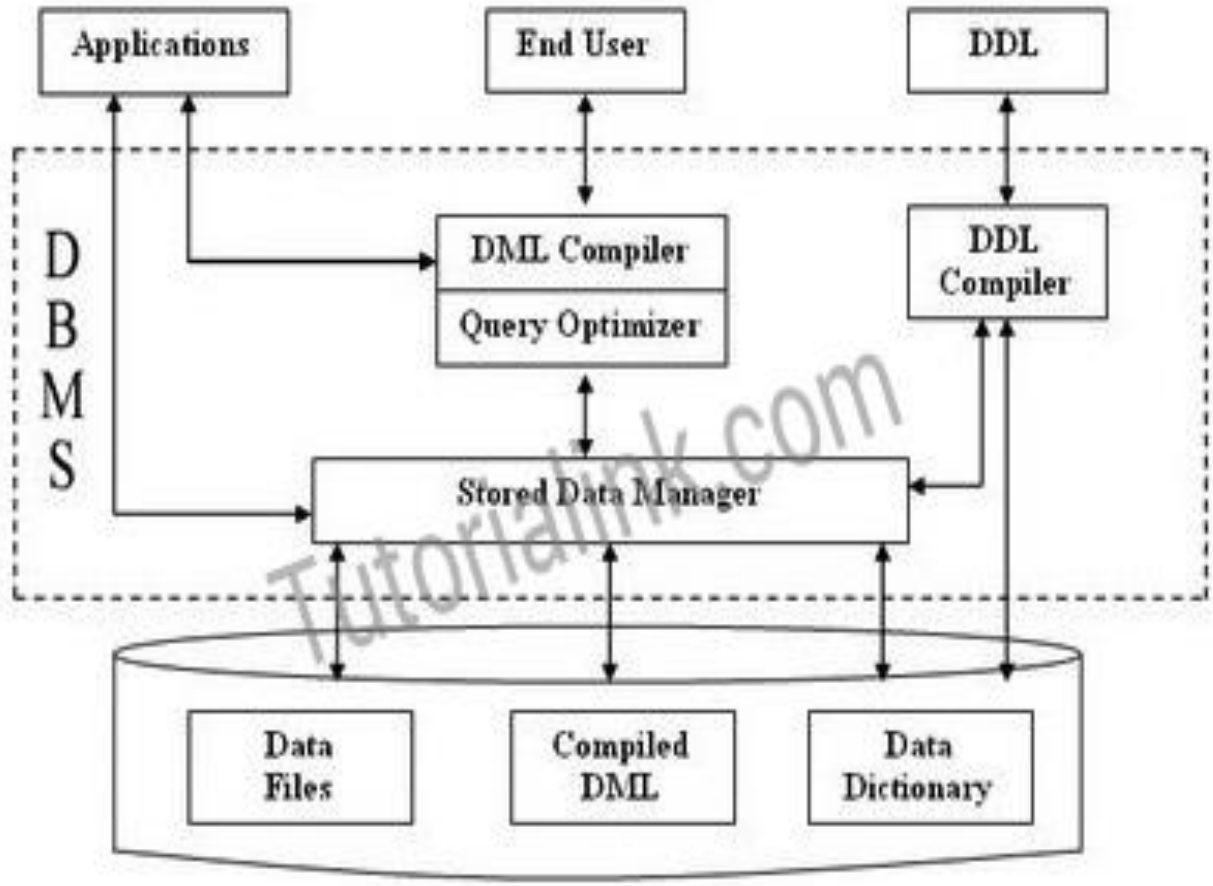
- **Database (Data) Tier** − At this tier, the database resides along with its query processing languages. We also have the relations that define the data and their constraints at this level.
- **Application (Middle) Tier** − At this tier reside the application server and the programs that access the database. For a user, this application tier presents an abstracted view of the database. End-users are unaware of any existence of the database beyond the application. At the other end, the database tier is not aware of any other user beyond the application tier. Hence, the application layer sits in the middle and acts as a mediator between the end-user and the database.
- **User (Presentation) Tier** − End-users operate on this tier and they know nothing about any existence of the database beyond this

layer. At this layer, multiple views of the database can be provided by the application. All views are generated by applications that reside in the application tier.

3. DBMS Component Modules and structure:

Structure of DBMS:

- DBMS (Database Management System) acts as an interface between the user and the database. The user requests the DBMS to perform various operations such as insert, delete, update and retrieval on the database.
- The components of DBMS perform these requested operations on the database and provide necessary data to the users.



- The various components of DBMS are described below:

Components of a DBMS

- The components of DBMS can be divided into two parts:

Function and Services of DBMS:

1. **DDL Compiler:**
    - Data Description Language compiler processes schema definitions specified in the DDL.
    - It includes metadata information such as the name of the files, data items, storage details of each file, mapping information and constraints etc.
2. **DML Compiler and Query optimizer:**
    - The DML commands such as insert, update, delete, retrieve from the application program are sent to the DML compiler for compilation into object code for database access.
    - The object code is then optimized in the best way to execute a query by the query optimizer and then send to the data manager.
3. **Data Manager:**
    - The Data Manager is the central software component of the DBMS also knows as Database Control System.
    - The Main Functions Of Data Manager Are:
        1. Convert operations in user's Queries coming from the application programs or combination of DML Compiler and Query optimizer which is known as Query Processor from user's logical view to physical file system.
        2. Controls DBMS information access that is stored on disk.

3. It also controls handling buffers in main memory.
4. It also enforces constraints to maintain consistency and integrity of the data.
5. It also synchronizes the simultaneous operations performed by the concurrent users.
6. It also controls the backup and recovery operations.

4. **Data Dictionary**:
    - Data Dictionary, which stores metadata about the database, in particular the schema of the database.
    - names of the tables, names of attributes of each table, length of attributes, and number of rows in each table.
    - Detailed information on physical database design such as storage structure, access paths, files and record sizes.
    - Usage statistics such as frequency of query and transactions.
    - Data dictionary is used to actually control the data integrity, database operation and accuracy. It may be used as a important part of the DBMS
5. **Data Files:**
    - Which store the database itself.
6. **Compiled DML:**
    - The DML complier converts the high level Queries into low level file access commands known as compiled DML.
7. **End Users:**
    - The second class of users then is end user, who interacts with system from online workstation or terminals.
    - Use the interface provided as an integral part of the database system software.

3. User can request, in form of query, to access database either directly by using particular language, such as SQL, or by using some pre-developed application interface.
4. Such request are sent to query evaluation engine via DML pre-compiler and DML compiler
5. The query evaluation engine accepts the query and analyses it.
6. It finds the suitable way to execute the compiled SQL statements of the query.
7. Finally, the compiled SQL statements are executed to perform the specified operation

8. Query Processor Units:

Interprets DDL statements into a set of tables containing metadata.
Translates DML statements into low level instructions that the query evaluation engine understands.
Converts DML statements embedded in an application program into procedure calls int he host language.
Executes low level instructions generated by DML compiler.

a. DDL Interpreter
b. DML Compiler
c. Embedded DML Pre-compiler
d. Query Evalution Engine

9. Storage Manager Units

Checks the authority of users to access data.
Checks for the satisfaction of the integrity constraints.
Preserves atomicity and controls concurrency.
Manages allocation of splace on disk.

Fetches data from disk storage to memory for being used.

a.Authorization Manager

b.Integrity Manager

c.Transaction Manager

d.File manager

e.Buffer Manager

4. Three Levels Database Architecture:

Following are the three levels of database architecture,

1. Physical Level
2. Conceptual Level
3. External Level

three levels database architecture

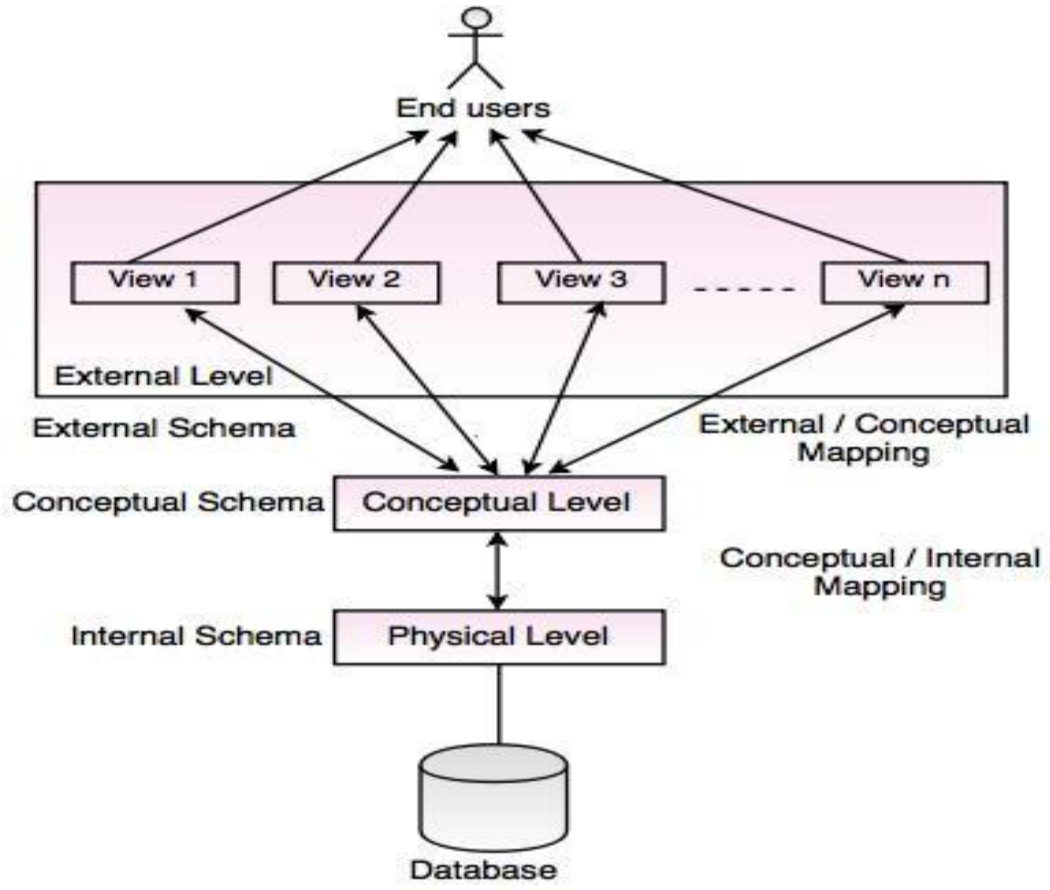


Fig. Three Level Architechture of DBMS

In the above diagram,

It shows the architecture of DBMS.

Mapping is the process of transforming request response between various database levels of architecture.

Mapping is not good for small database, because it takes more time.

In External / Conceptual mapping, DBMS transforms a request on an external schema against the conceptual schema.

In Conceptual / Internal mapping, it is necessary to transform the request from the conceptual to internal levels.

1**. Physical Level:**

Physical level describes the physical storage structure of data in database.
It is also known as Internal Level.
This level is very close to physical storage of data.
At lowest level, it is stored in the form of bits with the physical addresses on the secondary storage device.
At highest level, it can be viewed in the form of files.
The internal schema defines the various stored data types. It uses a physical data model.

2. **Conceptual Level:**

Conceptual level describes the structure of the whole database for a group of users.
It is also called as the data model.
Conceptual schema is a representation of the entire content of the database.
These schema contains all the information to build relevant external records.
It hides the internal details of physical storage.

3. **External Level:**

External level is related to the data which is viewed by individual end users.
This level includes a no. of user views or external schemas.
This level is closest to the user.
External view describes the segment of the database that is required for a particular user group and hides the rest of the database from that user group.

5.Schema and Instance:

- The data which is stored in the database at a particular moment of time is called an instance of the database.
- The overall design of a database is called schema.
- A database schema is the skeleton structure of the database. It represents the logical view of the entire database.
- A schema contains schema objects like table, foreign key, primary key, views, columns, data types, stored procedure, etc.
- A schema diagram can display only some aspects of a schema like the name of record type, data type, and constraints.
- Example:

STUDENT

Name	Student_number	Class	Major

COURSE

Course_name	Course_number	Credit_hours	Department

PREREQUISITE

Course_number	Prerequisite_number

SECTION

Section_identifier	Course_number	Semester	Year	Instructor

GRADE_REPORT

Student_number	Section_identifier	Grade

6.Data Independence:

One of the biggest advantages of database is data independence. It means we can change the conceptual schema at one level without affecting the data at other level. It means we can change the structure of a database without affecting the data required by users and program. This feature was not available in file oriented approach. There are two types of data independence and they are:

1. Physical data independence
2. Logical data independence

Data Independence The ability to modify schema definition in on level without affecting schema definition in the next higher level is called data independence. There are two levels of data independence:

1. **Physical data independence** is the ability to modify the physical schema without causing application programs to be rewritten. Modifications at the physical level are occasionally necessary to improve performance. It means we change the physical storage/level without affecting the conceptual or external view of the data. The new changes are absorbed by mapping techniques.

2. **Logical data independence** in the ability to modify the logical schema without causing application program to be rewritten. Modifications at the logical level are necessary whenever the logical structure of the database is altered (for example, when money-market accounts are added to banking system).

Logical Data independence means if we add some new columns or remove some columns from table then the user view and programs should not changes. It is called the logical independence. For example: consider two users A & B. Both are selecting the empno and ename. If user B add a new column salary in his view/table then it will not effect the external view user; user A, but internal view of database has been changed for both users A & B. Now user A can also print the salary.

7.Data Models:

A Database model defines the logical design and structure of a database and defines how data will be stored, accessed and updated in a database management system. While the **Relational Model** is the most widely used database model, there are other models too:

- Hierarchical Model
- Network Model
- Entity-relationship Model
- Relational Model

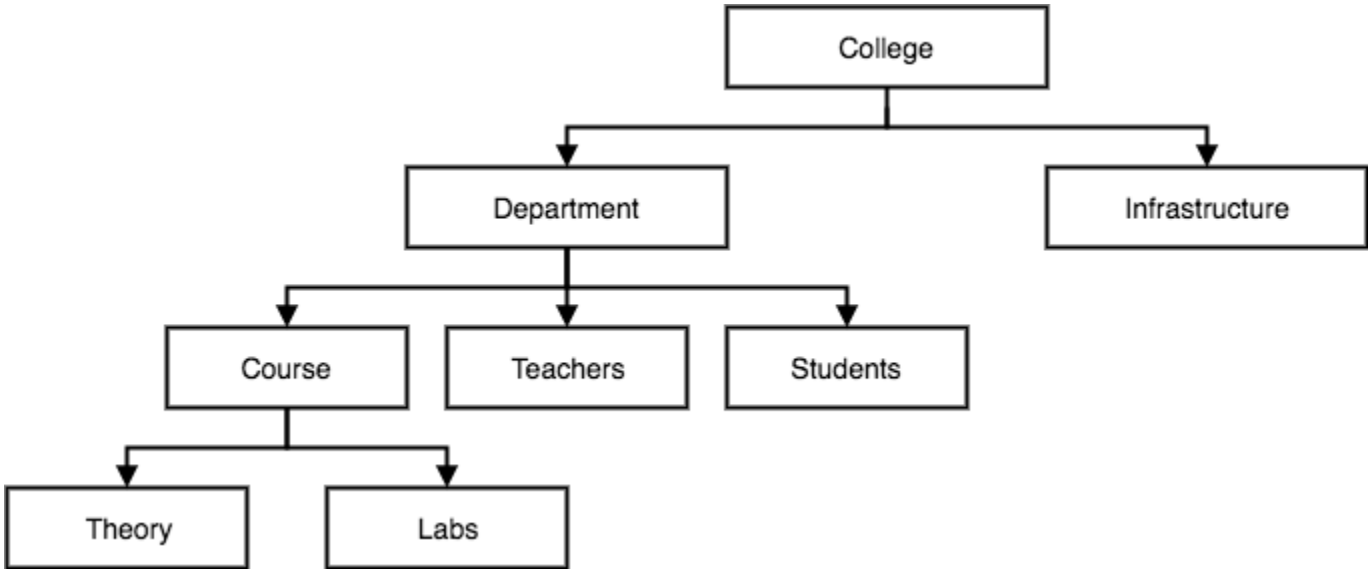
Hierarchical Model

This database model organises data into a tree-like-structure, with a single root, to which all the other data is linked. The heirarchy starts from the **Root** data, and expands like a tree, adding child nodes to the parent nodes.

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.

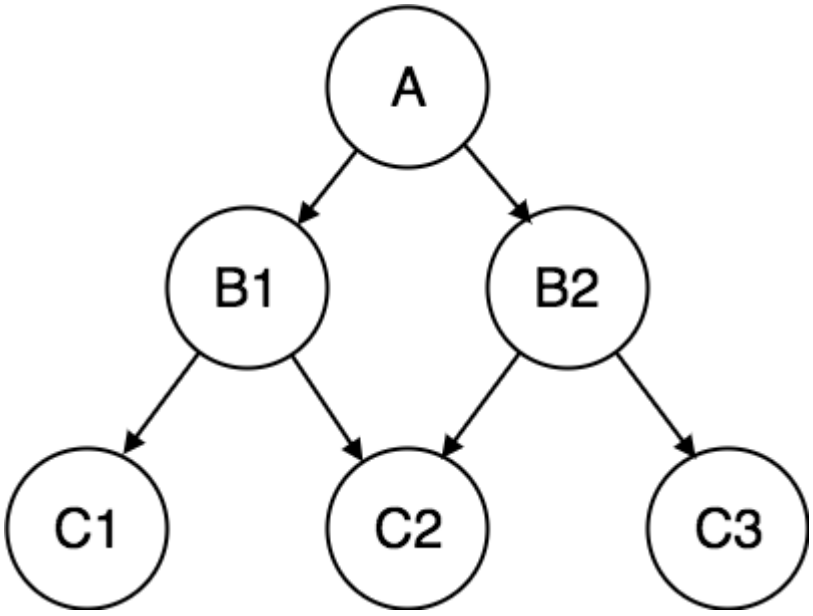


Network Model

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.



Entity-relationship Model

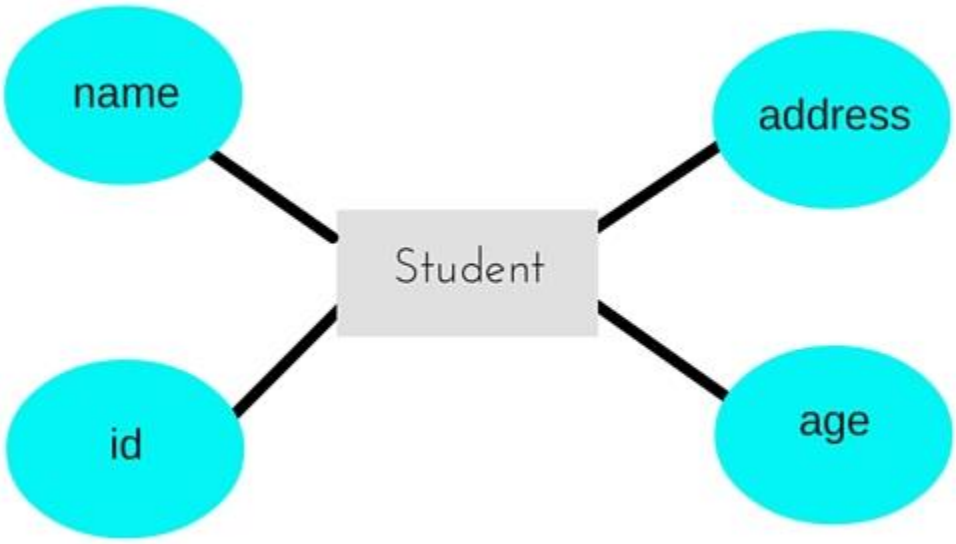
In this database model, relationships are created by dividing object of interest into entity and its characteristics into attributes.

Different entities are related using relationships.

E-R Models are defined to represent the relationships into pictorial form to make it easier for different stakeholders to understand.

This model is good to design a database, which can then be turned into tables in relational model(explained below).

Let's take an example, If we have to design a School Database,
then **Student** will be an **entity** with **attributes** name, age, address etc.
As **Address** is generally complex, it can be
another **entity** with **attributes** street name, pincode, city etc, and there will be a relationship between them.



Relational Model

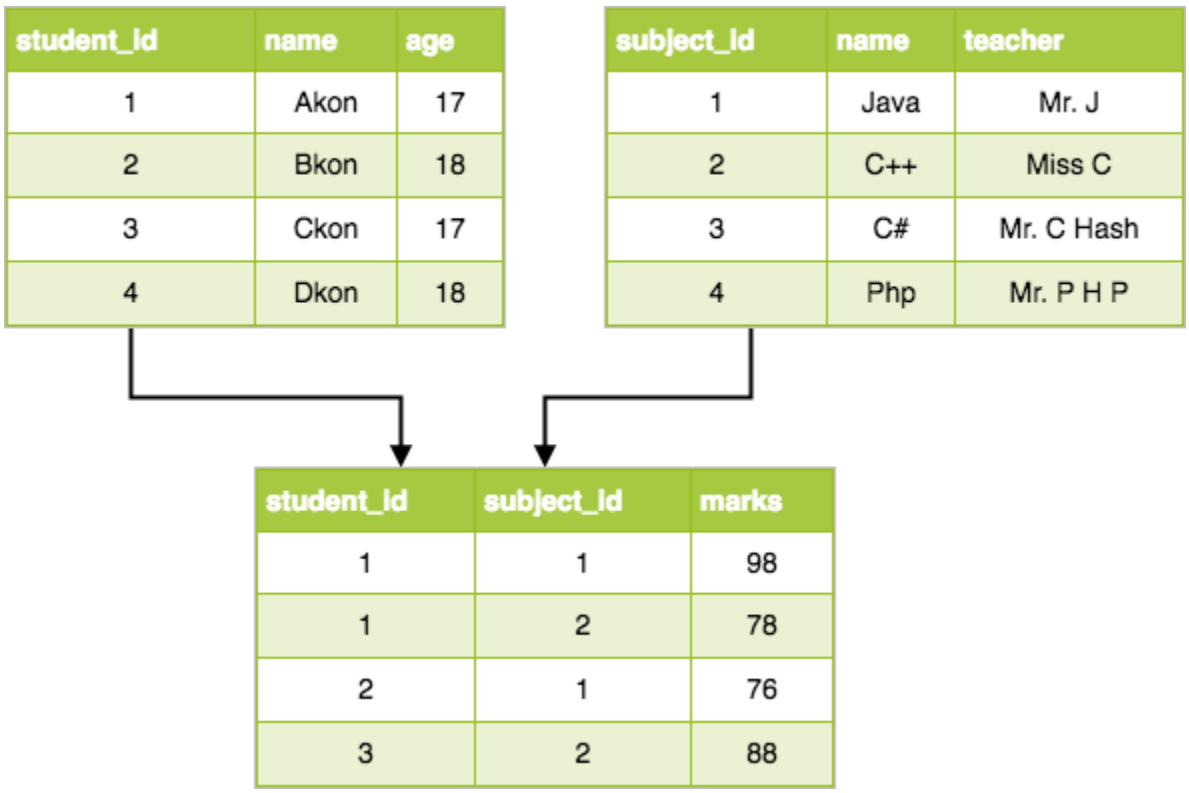
In this model, data is organised in two-dimensional **tables** and the relationship is maintained by storing a common field.

This model was introduced by E.F Codd in 1970, and since then it has been the most widely used database model, infact, 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

8. All Database languages:

Data Definition Language (DDL):

statements are used to classify the database structure or schema. It is a type of language that allows the DBA or user to depict and name those entities, attributes, and relationships that are required for the application along with any associated integrity and security constraints. Here are the lists of tasks that come under DDL:

• CREATE - used to create objects in the database.

• ALTER - used to alters the structure of the database.

• DROP - used to delete objects from the database.

• TRUNCATE - used to remove all records from a table, including all spaces allocated for the records are removed.

• COMMENT - used to add comments to the data dictionary • RENAME - used to rename an object.

Data Manipulation Language:

A language that offers a set of operations to support the fundamental data manipulation operations on the data held in the database. Data Manipulation Language (DML) statements are used to manage data within schema objects. Here are the lists of tasks that come under DML:

• SELECT - It retrieves data from a database.

• INSERT - It inserts data into a table.

• UPDATE - It updates existing data within a table.

• DELETE - It deletes all records from a table, the space for the records remain.

• MERGE - UPSERT operation (insert or update).

• CALL - It calls a PL/SQL or Java subprogram.

• EXPLAIN PLAN - It explains access path to data.

• LOCK TABLE - It controls concurrency.

Data Control Language:

There are another two forms of database sub-languages. The Data Control Language (DCL) is used to control privilege in Database. To perform any operation in the database, such as for creating tables, sequences or views we need privileges. Privileges are of two types,

• System - creating a session, table, etc. are all types of system privilege.

• Object - any command or query to work on tables comes under object privilege. DCL is used to define two commands. These are:

• Grant - It gives user access privileges to a database.

• Revoke - It takes back permissions from the user.

Transaction Control Language (TCL):

Transaction Control statements are used to run the changes made by DML statements. It allows statements to be grouped into logical transactions.

• COMMIT - It saves the work done.

• SAVEPOINT - It identifies a point in a transaction to which you can later roll back.

• ROLLBACK - It restores the database to original since the last COMMIT.

• SET TRANSACTION - It changes the transaction options like isolation level and what rollback segment to use.

9.Data Modeling using the Entity Relationship Model:

Entity-Relationship Model:

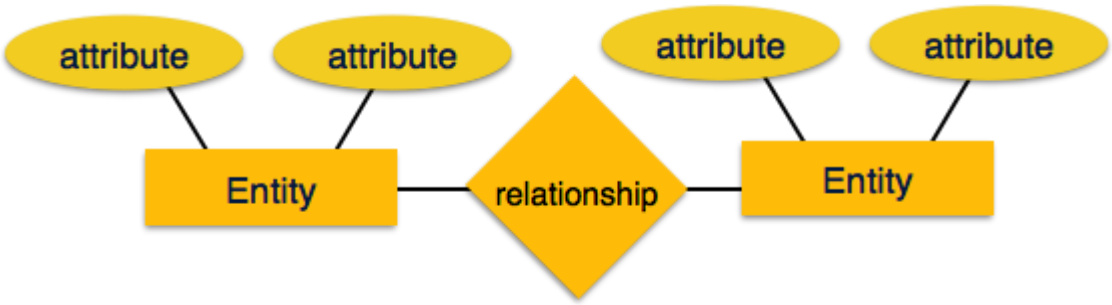
Entity-Relationship (ER) Model is based on the notion of real-world entities and relationships among them. While formulating real-world scenario into the database model, the ER Model creates entity set, relationship set, general attributes and constraints.

ER Model is best used for the conceptual design of a database.

ER Model is based on −

- Entities and their *attributes.*
- Relationships among entities.

These concepts are explained below**.**



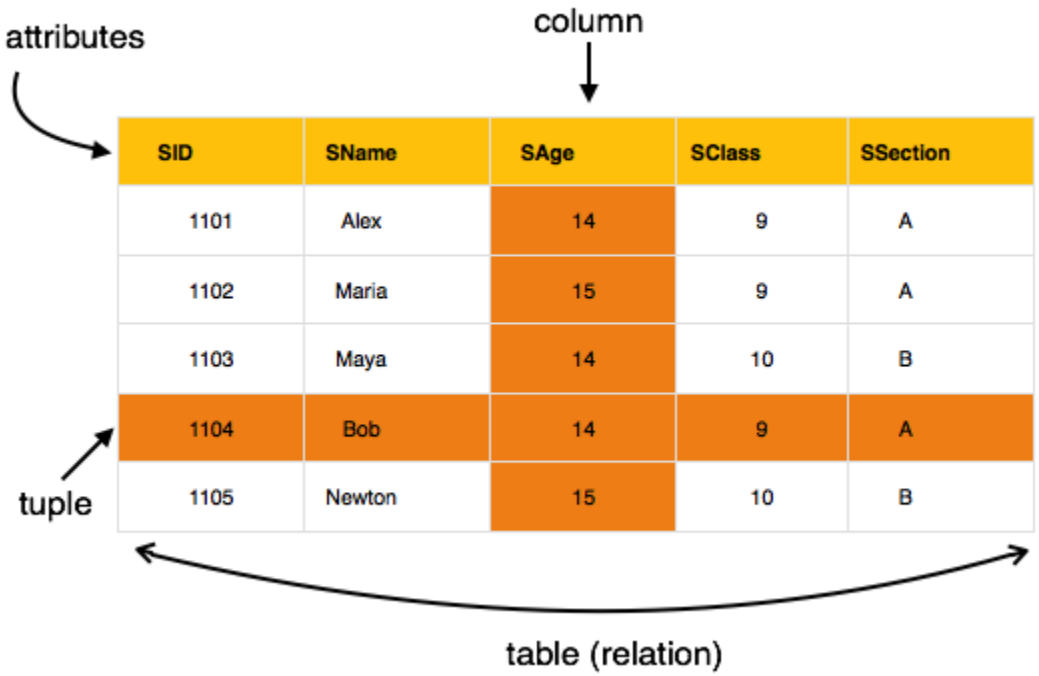
- **Entity** − An entity in an ER Model is a real-world entity having properties called attributes. Every attribute is defined by its set of values called domain. For example, in a school database, a student is considered as an entity. Student has various attributes like name, age, class, etc.
- **Relationship** − The logical association among entities is called *relationship*. Relationships are mapped with entities in various ways. Mapping cardinalities define the number of association between two entities.

Mapping cardinalities −

- **one to one**
- **one to many**
- **many to one**
- **many to many**

Relational Model

The most popular data model in DBMS is the Relational Model. It is more scientific a model than others. This model is based on first-order predicate logic and defines a table as an n-ary relation.



The main highlights of this model are −

- Data is stored in tables called relations.
- Relations can be normalized.
- In normalized relations, values saved are atomic values.
- Each row in a relation contains a unique value.
- Each column in a relation contains values from a same domain.

10.ER model concepts:

The ER model defines the conceptual view of a database. It works around real-world entities and the associations among them. At view level, the ER model is considered a good option for designing databases.

Entity:

An **entity** can be a real-world object, either animate or inanimate, that can be easily identifiable. For example, in a school database, students, teachers, classes, and courses offered can be considered as entities. All

these entities have some attributes or properties that give them their identity.

An entity set is a collection of similar types of entities. An entity set may contain entities with attribute sharing similar values. For example, a Students set may contain all the students of a school; likewise a Teachers set may contain all the teachers of a school from all faculties. Entity sets need not be disjoint.

Attributes:

Entities are represented by means of their properties, called attributes. All attributes have values. For example, a student entity may have name, class, and age as attributes.

There exists a domain or range of values that can be assigned to attributes. For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc.

Types of Attributes

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

- **Multi-value attribute** − Multi-value attributes may contain more than one values. For example, a person can have more than one phone number, email_address, etc.

These attribute types can come together in a way like −

- simple single-valued attributes
- simple multi-valued attributes
- composite single-valued attributes
- composite multi-valued attributes

Entity-Set and Keys:

Key is an attribute or collection of attributes that uniquely identifies an entity among entity set.

For example, the roll_number of a student makes him/her identifiable among students.

- **Super Key** − A set of attributes (one or more) that collectively identifies an entity in an entity set.
- **Candidate Key** − A minimal super key is called a candidate key. An entity set may have more than one candidate key.
- **Primary Key** − A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.

Relationship:

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

Relationship Set:

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

Degree of Relationship:

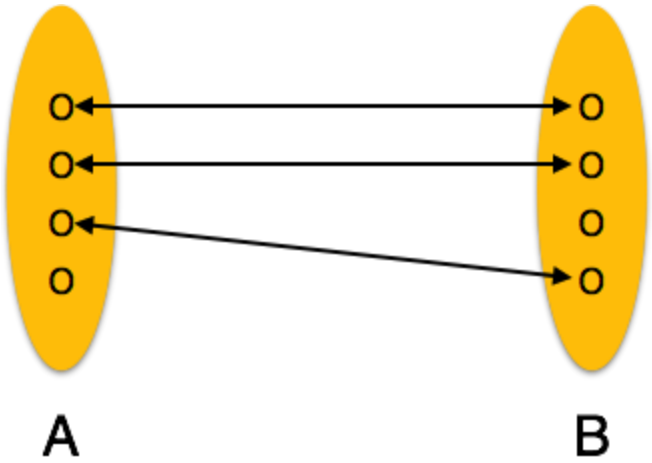
The number of participating entities in a relationship defines the degree of the relationship.

- Binary = degree 2
- Ternary = degree 3
- n-ary = degree

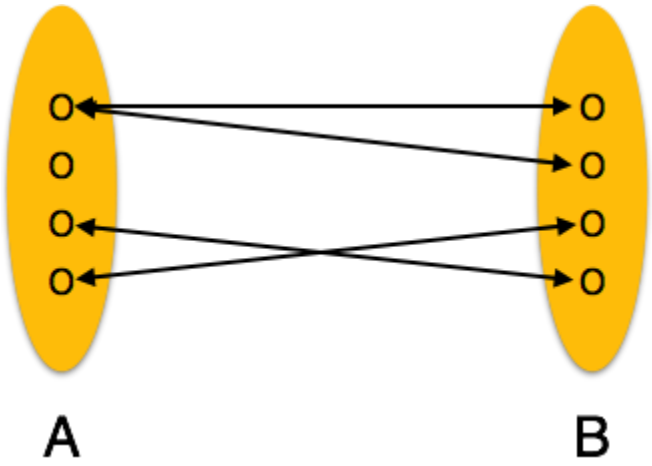
Mapping Cardinalities:

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

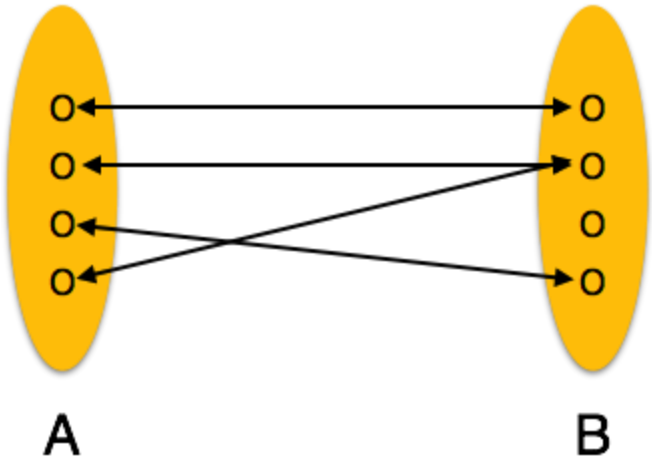
- One-to-one − One entity from entity set A can be associated with at most one entity of entity set B and vice versa.



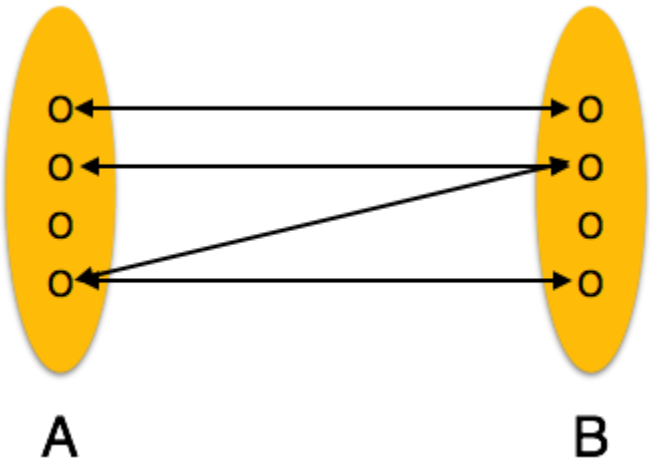
- **One-to-many** − One entity from entity set A can be associated with more than one entities of entity set B however an entity from entity set B, can be associated with at most one entity.



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



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



11.ER Diagram Representation:

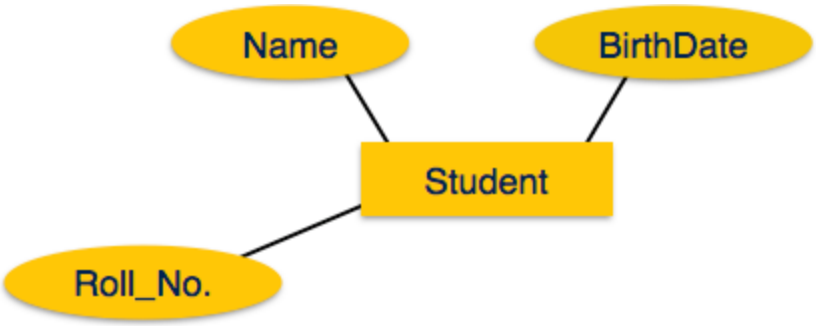
Any object, for example, entities, attributes of an entity, relationship sets, and attributes of relationship sets, can be represented with the help of an ER diagram.

Entity

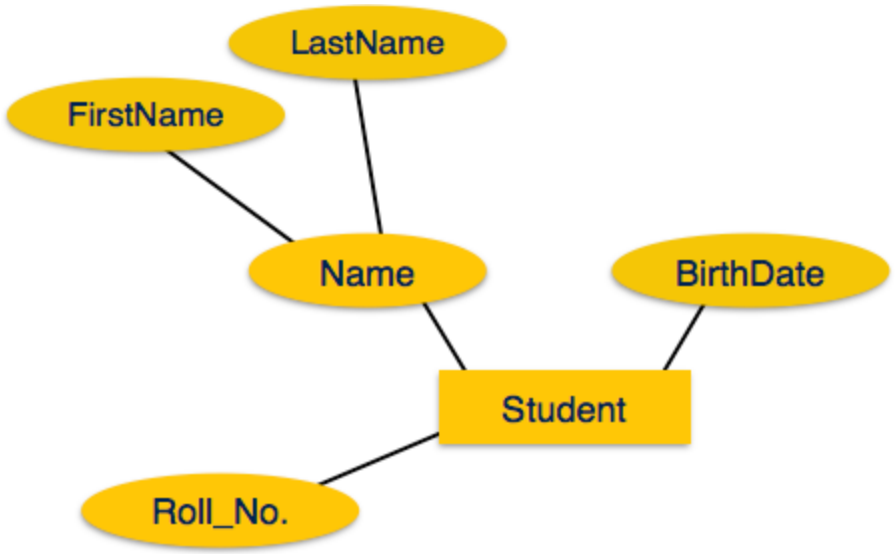
Entities are represented by means of rectangles. Rectangles are named with the entity set they represent.

Attributes

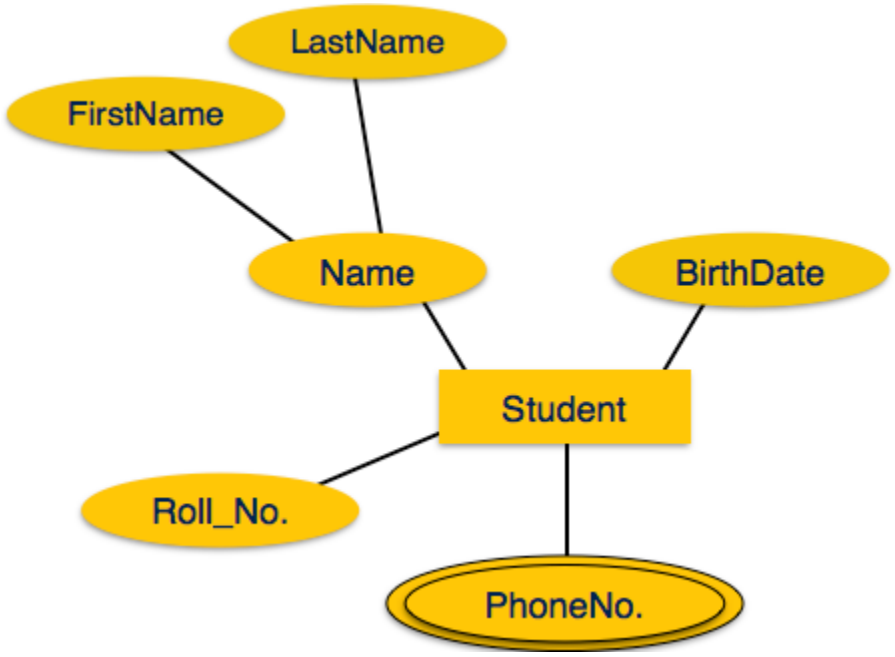
Attributes are the properties of entities. Attributes are represented by means of ellipses. Every ellipse represents one attribute and is directly connected to its entity (rectangle).



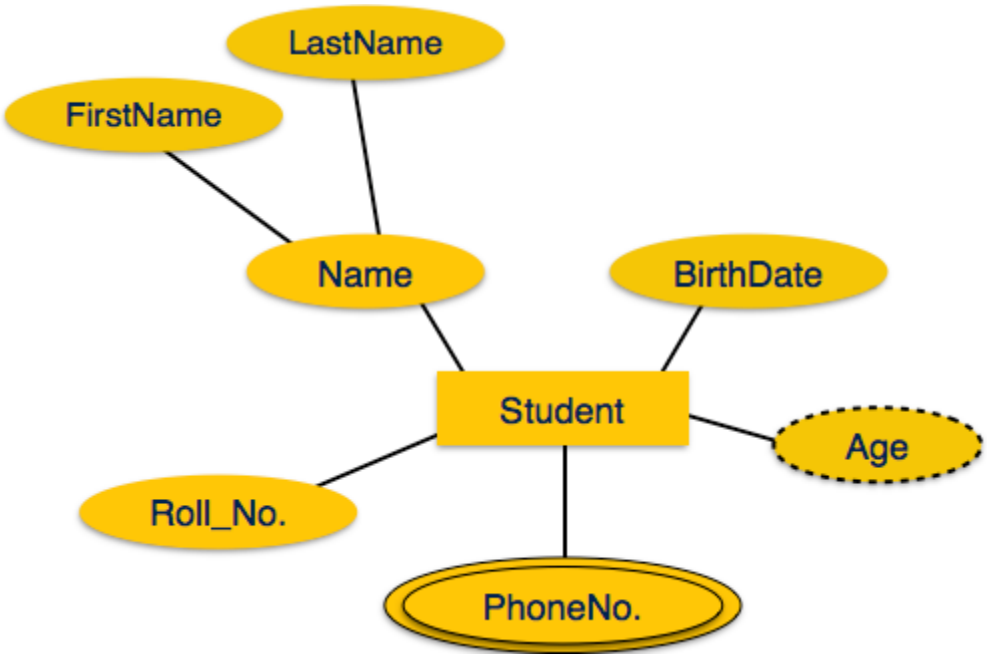
If the attributes are **composite**, they are further divided in a tree like structure. Every node is then connected to its attribute. That is, composite attributes are represented by ellipses that are connected with an ellipse.



Multivalued attributes are depicted by double ellipse.



Derived attributes are depicted by dashed ellipse.



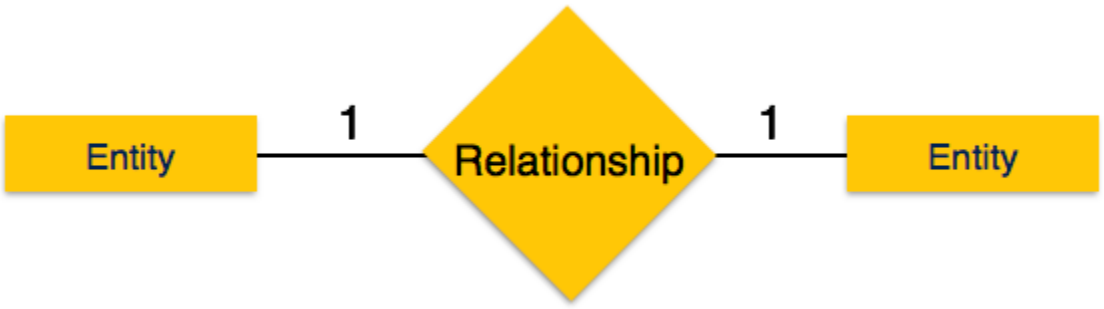
Relationship:

Relationships are represented by diamond-shaped box. Name of the relationship is written inside the diamond-box. All the entities (rectangles) participating in a relationship, are connected to it by a line.

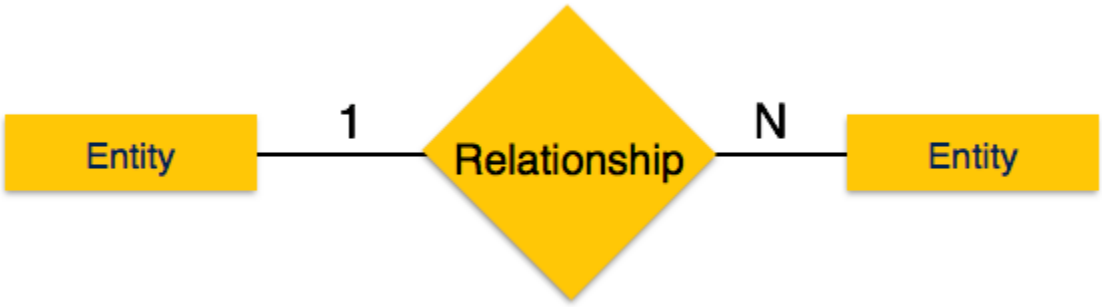
Binary Relationship and Cardinality:

A relationship where two entities are participating is called a **binary relationship**. Cardinality is the number of instance of an entity from a relation that can be associated with the relation.

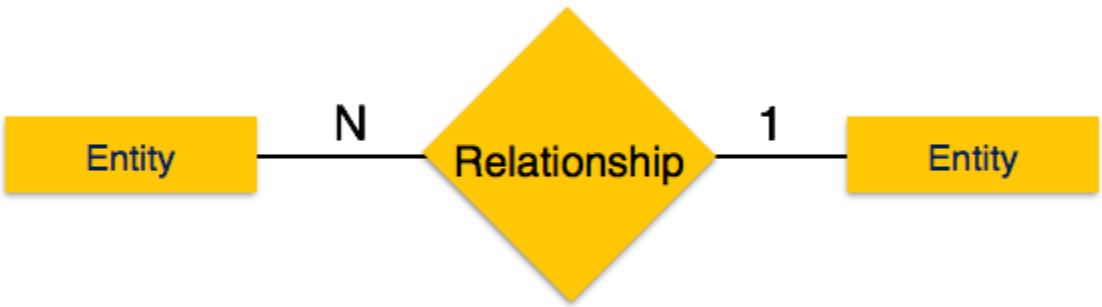
- **One-to-one** − When only one instance of an entity is associated with the relationship, it is marked as '1:1'. The following image reflects that only one instance of each entity should be associated with the relationship. It depicts one-to-one relationship.



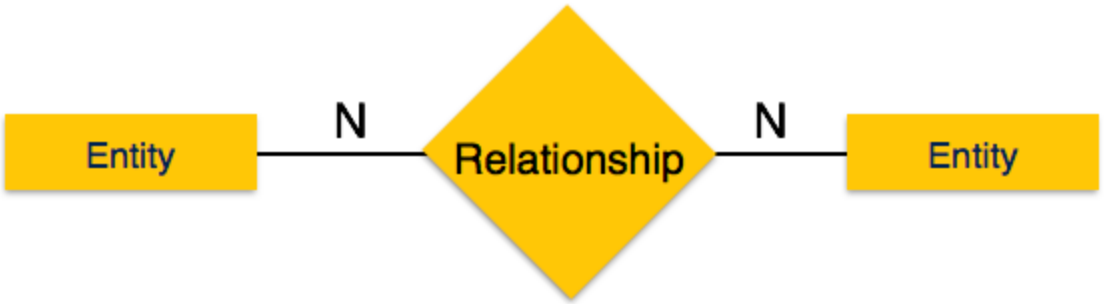
- **One-to-many** − When more than one instance of an entity is associated with a relationship, it is marked as '1:N'. The following image reflects that only one instance of entity on the left and more than one instance of an entity on the right can be associated with the relationship. It depicts one-to-many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



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

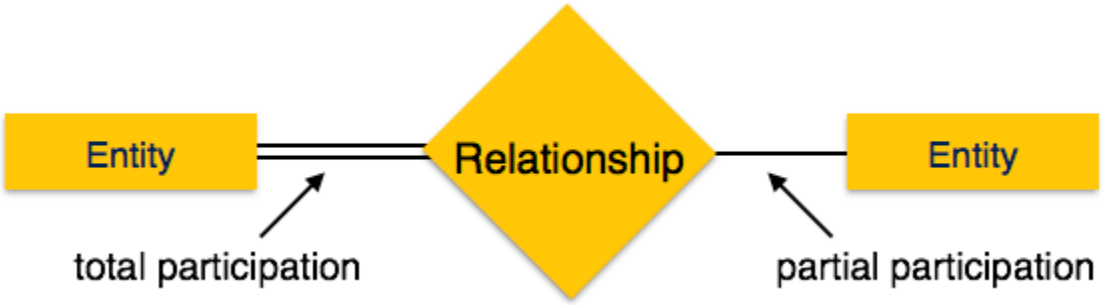


- **Many-to-many** − The following image reflects that more than one instance of an entity on the left and more than one instance of an entity on the right can be associated with the relationship. It depicts many-to-many relationship.



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.



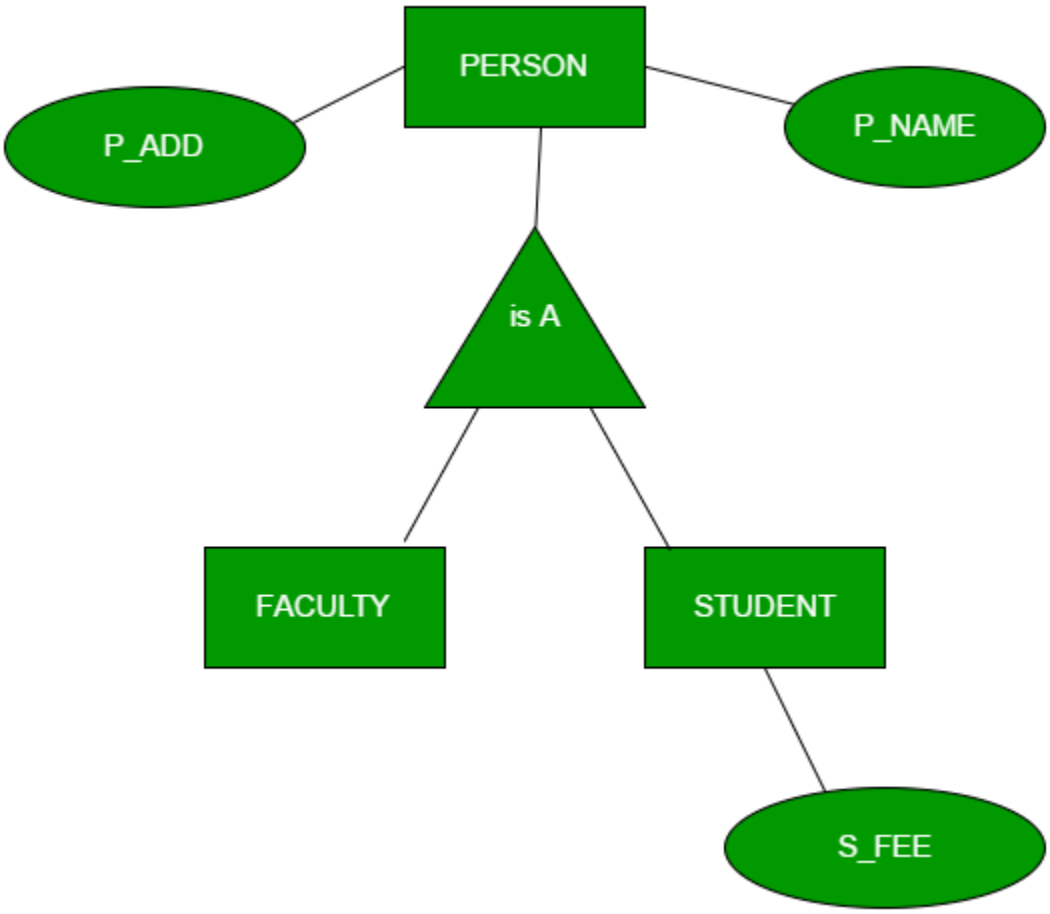
12.Generalization,Specialization & Aggregration:

Generalization, Specialization and Aggregation in ER model are used for data abstraction in which abstraction mechanism is used to hide details of a set of objects.

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

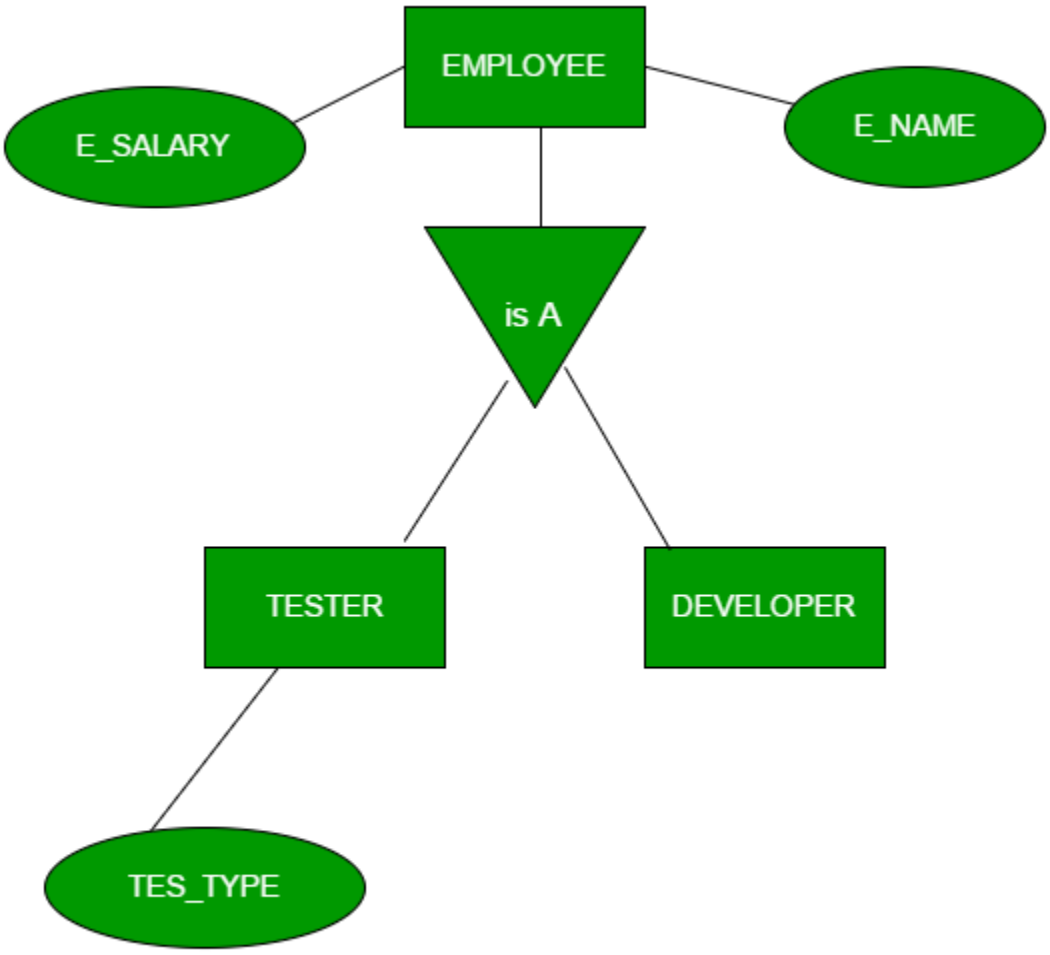
common. For Example, STUDENT and FACULTY can be generalized to a higher level entity called PERSON as shown in Figure . In this case, common attributes like P_NAME, P_ADD become part of higher entity (PERSON) and specialized attributes like S_FEE become part of specialized entity (STUDENT).



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 . 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).

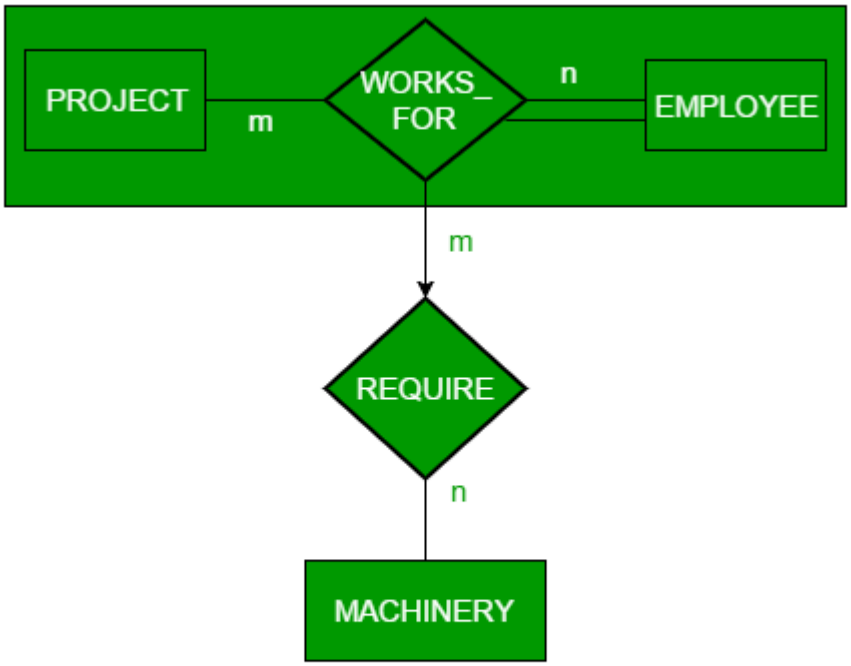


Specialization

Aggregation –

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. For Example, Employee working for a project may require some machinery. So, REQUIRE relationship is needed between

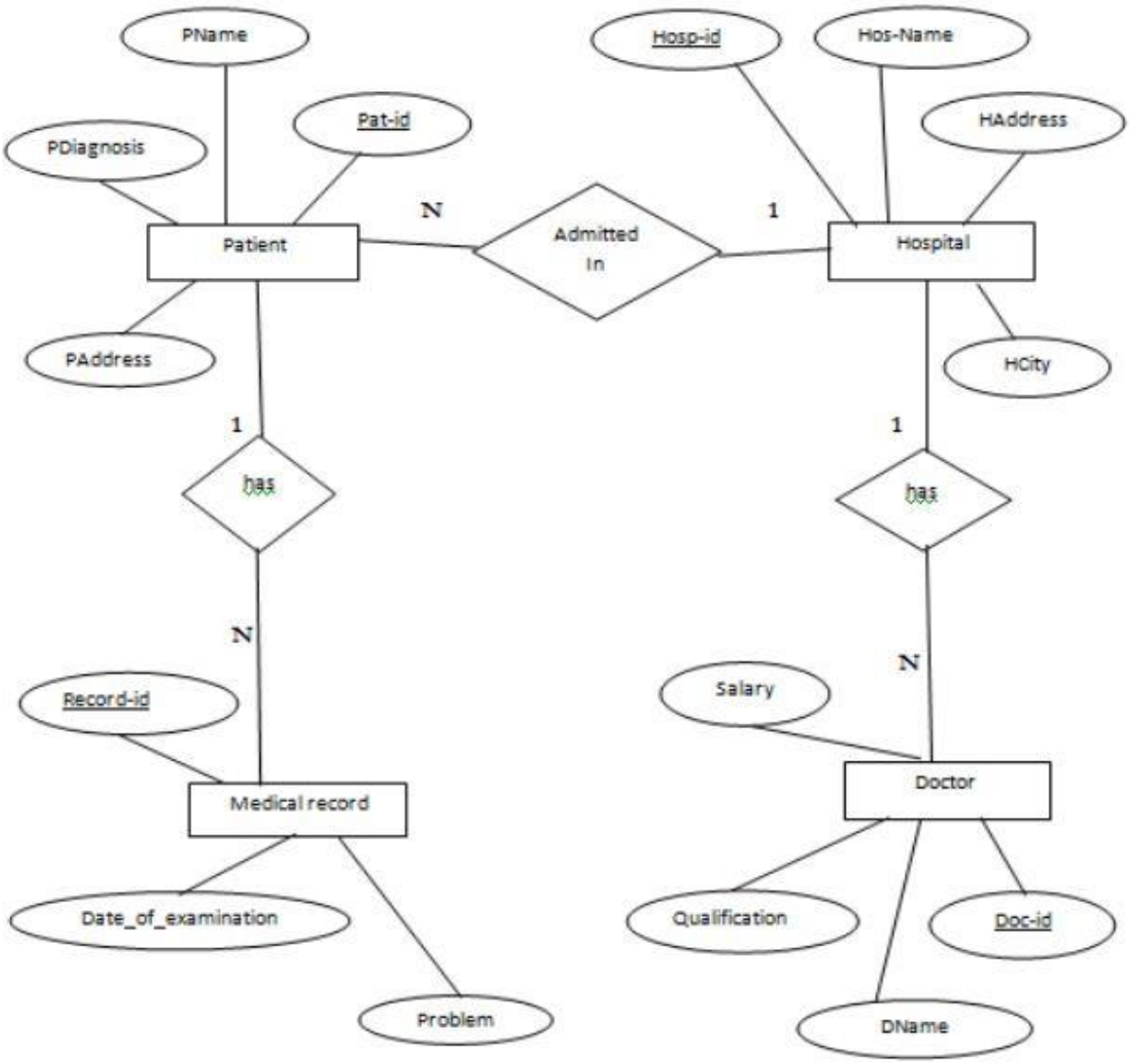
relationship WORKS_FOR and entity MACHINERY. Using aggregation, WORKS_FOR relationship with its entities EMPLOYEE and PROJECT is aggregated into single entity and relationship REQUIRE is created between aggregated entity and MACHINERY.



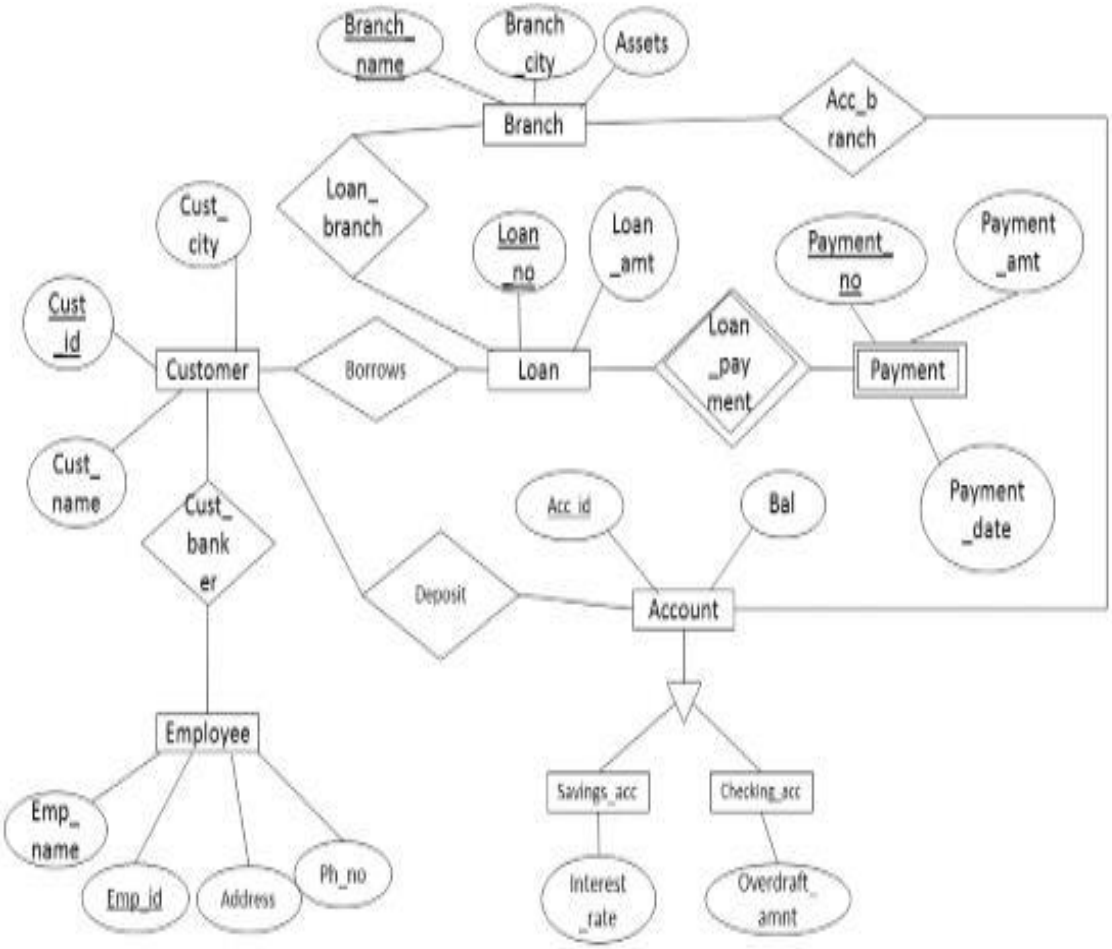
Aggregation

13.Examples of ER Daigram:

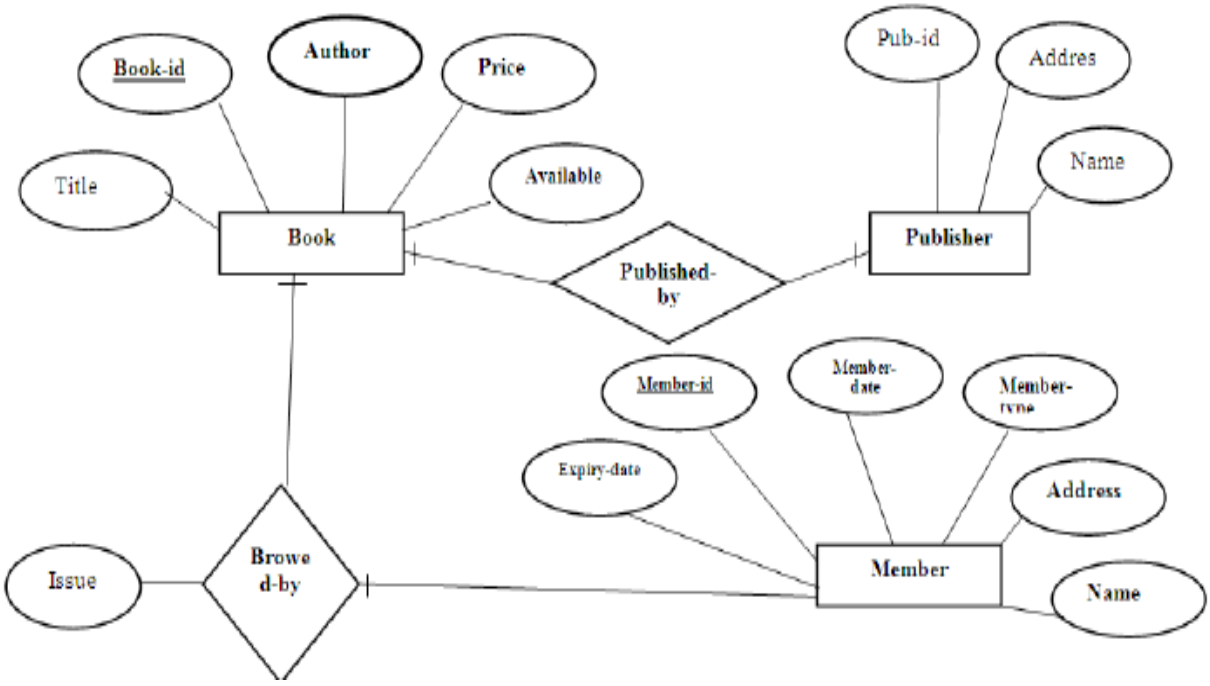
1.Hospital Management System:



2.Bank Management System:



3.Library Management System:


Book-id
Author
Price
Pub-id
Addres
Title
Available
Name
Book
Publisher
Published-by
Member-id
Member-date
Member-type
Expiry-date
Address
Issue
Browe d-by
Member
Name

THANK YOU