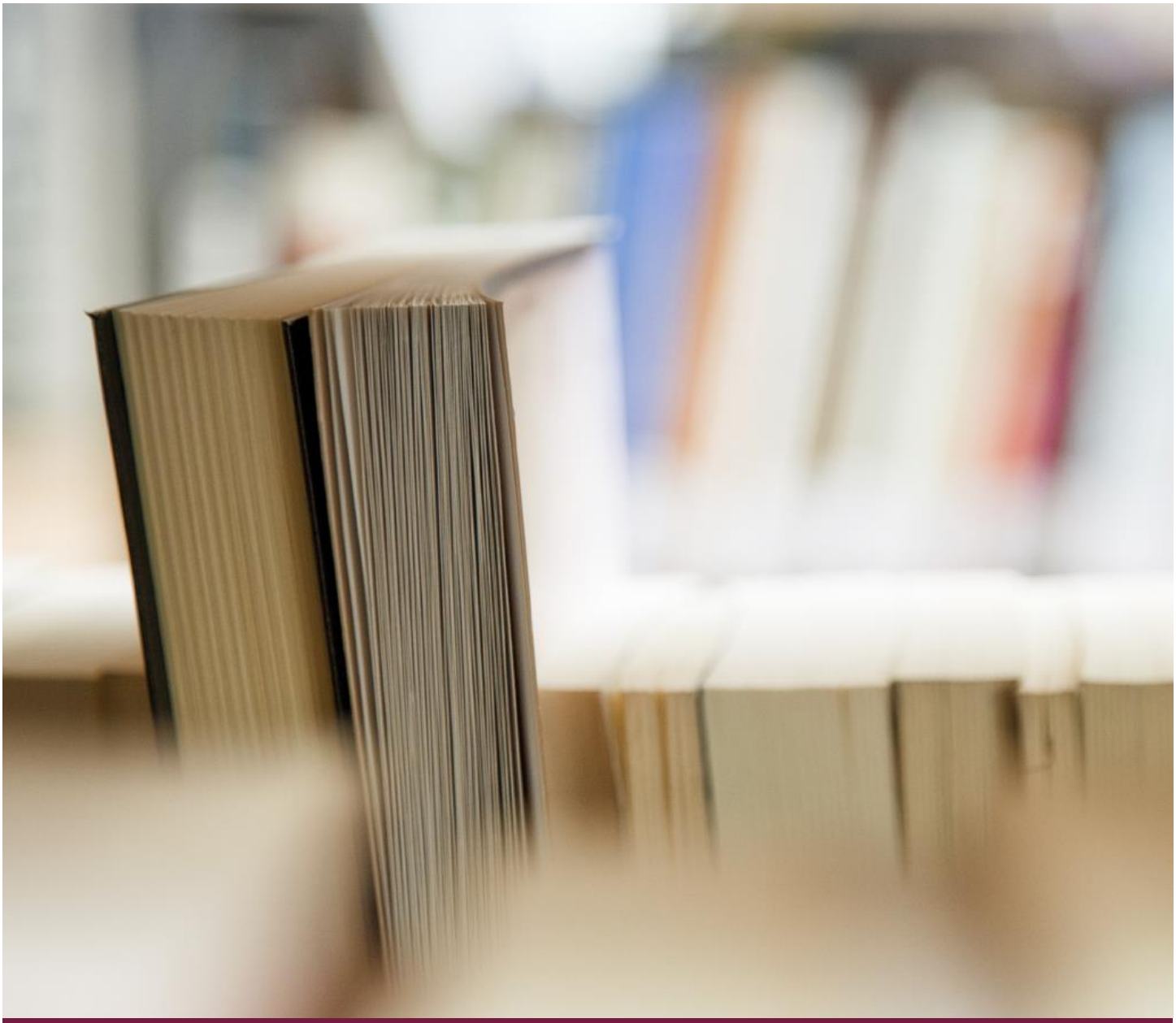


E-R MODELS

SYKKSU

Prof. Akshara Chavan

UNIT-02

E-R MODELS

DATA MODELLING

- A data model is a picture or description which shows how the data is to be arranged to achieve a given task.
- It is a clear model which specifies how the data items are arranged in a given model.
- Some data models which gives a clear picture which shows the manner in which the data records are connected or related within a file structure. These are called structural data models.
- DBMS organize and structure data so that it can be retrieved and manipulated by different users and application programs.
- The data structures and access techniques provided by a particular DBMS are called its data model.
- A data model determined both the personality of a DBMS and the applications for which it is particularly well suited.

Advantages

1. Simplicity: Since the database is based on the hierarchical structure, the relationship between the various layers is logically simple.

2. Data Security: Hierarchical model was the first database model that offered the data security that is provided by the DBMS.

3. Data Integrity: Since it is based on the parent child relationship, there is always a link between the parent segment and the child segment under it.

4. Efficiency: It is very efficient because when the database contains a large number of 1:N relationship and when the user require large number of transaction.

Disadvantages

1. Implementation complexity: Although it is simple and easy to design, it is quite complex to implement.

2. Database Management Problem: If you make any changes in the database structure, then you need to make changes in the entire application program that access the database.

3. Lack of Structural Independence: there is lack of structural independence because when we change the structure then it becomes compulsory to change the application too.

4. Operational Anomalies: Hierarchical model suffers from the insert, delete and update anomalies, also retrieval operation is difficult.

BASIC BUILDING BLOCKS

The basic building blocks of all data models are entities, attributes, relationships, and constraints.

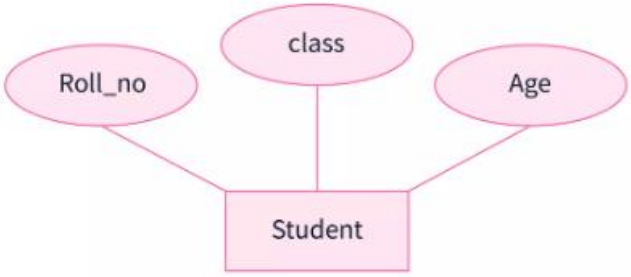
ENTITY:

- An entity is anything (a person, a place, a thing, or an event) about which data are to be collected and stored.
- An entity represents a particular type of object in the real world. Because an entity represents a particular type of object, entities are —distinguishable‖—that is, each entity occurrence is unique and distinct.
- For example, a CUSTOMER entity would have many distinguishable customer occurrences, such as ajay, pravin, aniket, etc. Entities may be physical objects, such as customers or products, but entities may also be abstractions, such as flight routes or musical concerts.
- **Strong Entity**
  - A strong entity is not dependent on 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. Various strong entities, when combined together, create a strong entity set.
- **Weak Entity**
  - A weak entity is dependent on a strong entity to ensure its existence. Unlike a strong entity, a weak entity does not have any primary 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. This relationship is also known as identifying relationship.

ATTRIBUTE:

- An attribute is a characteristic of an entity.
- For example, a CUSTOMER entity would be described by attributes such as customer last name, customer first name, customer phone, customer address, and customer credit limit.
- Attributes are the equivalent of fields in file systems.
- An entity is represented by a set of attributes, that is descriptive properties possessed by all members of an entity set
- Domain – the set of permitted values for each attribute type.

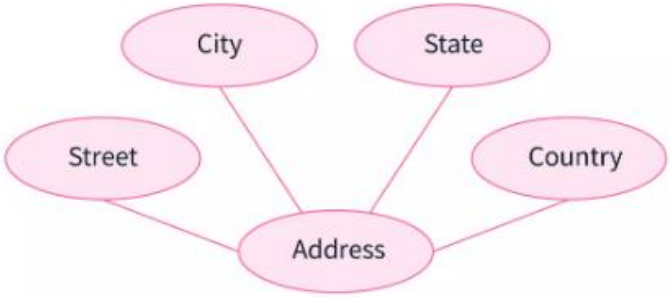
1. **Simple attributes**
   - Simple attributes in an ER model diagram are independent attributes that can't be classified further and also, can't be subdivided into any other component and have single value in it.



- As we can see in the above example, Student is an entity represented by a rectangle, and it consists of attributes: Roll_no, class, and Age. Also, there is a point to be noted that we can't further subdivide the Roll_no attribute and even the other two attributes into sub-attributes. Hence, they are known as simple attributes of the Student entity.

2. Composite Attributes

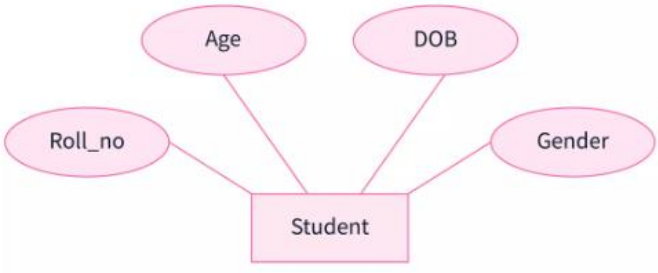
- Composite attributes have opposite functionality to that of simple attributes as we can further subdivide composite attributes into different components or sub-parts that form simple attributes.
- **In simple terms, composite attributes are composed of one or more simple attributes.**



- As we can see in the above example, Address **is a composite attribute** represented by an elliptical shape, and it can be further subdivided into many simple attributes like Street, City, State, Country, Landmark, etc.

3. Single-Valued Attributes

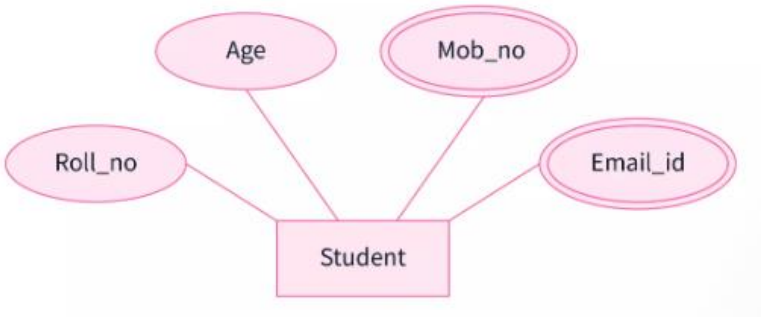
- Single valued attributes are those attributes that consist of a single value for each entity instance and can't store more than one value. The value of these single-valued attributes always remains the same, just like the name of a person.



- Each entity instance can have only one Roll_no, which is a unique, single DOB by which we can calculate age and also fixed gender. Also, we can't further subdivide these attributes, and hence, they are **simple as well as single-valued attributes.**

4. Multi-Valued Attributes

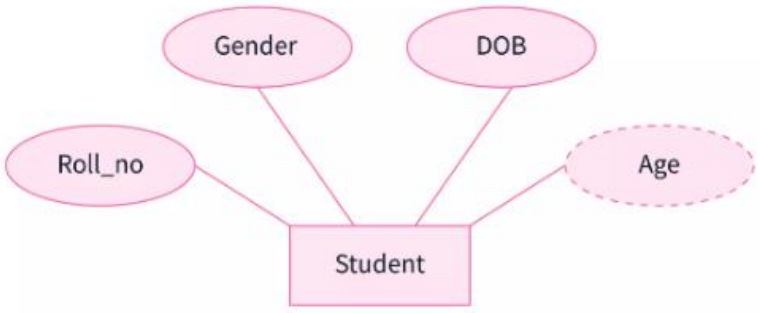
- Multi-valued attributes have opposite functionality to that of single-valued attributes, and as the name suggests, multi-valued attributes can take up and store more than one value at a time for an entity instance from a set of possible values. These attributes are represented by **co-centric elliptical shape** to represent multi-valued attributes inside it.



- As we can see in the above example, the Student entity has four attributes: Roll_no and Age are simple as well as single-valued attributes as discussed above but Mob_no and Email_id are represented by co-centric ellipse **are multi-valued attributes.**

5. Derived Attributes

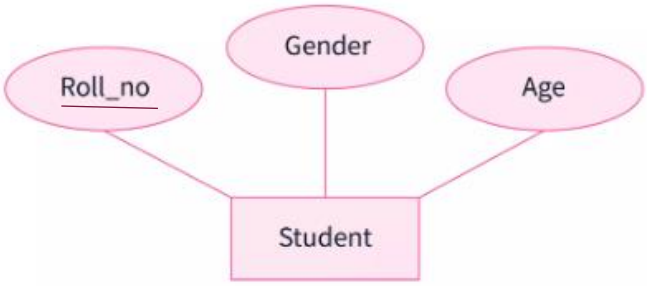
- Derived attributes are those attributes whose values can be derived from the values of other attributes. They are always dependent upon other attributes for their value.
- **For example,** As we were discussing above, DOB is a single-valued attribute and remains constant for an entity instance. From DOB, we can derive the Age attribute, which changes every year, and can easily calculate the age of a person from his/her date of birth value. Hence, the Age attribute here is derived attribute from the DOB single-valued attribute.



- **Derived attributes are always represented by dashed or dotted elliptical shapes.**

6. Key Attributes

- Key attributes are special types of attributes that act as the primary key for an entity and they can uniquely identify an entity from an entity set. The values that key attributes store must be unique and non-repeating.
-

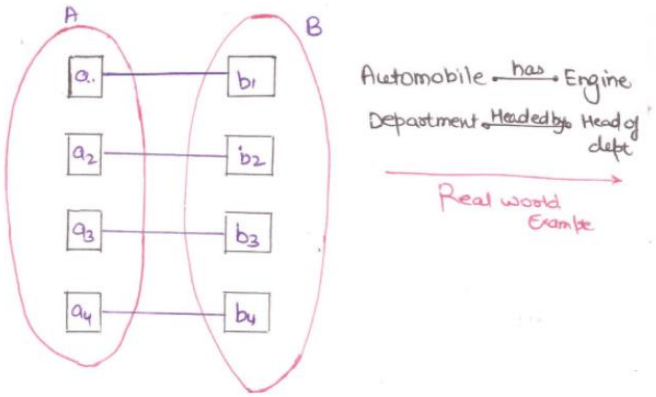


RELATIONSHIP:

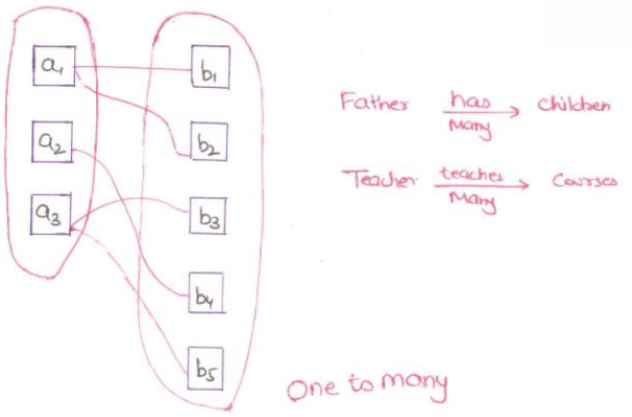
- A relationship describes an association among entities.
- For example, a relationship exists between customers and agents that can be described as follows: an agent can serve many customers, and each customer may be served by one agent.
- Data models use three types of relationships: **one-to-many, many-to-many, and one-to-one.**
- Database designers usually use the shorthand notations 1:M ,M:N and 1:1 respectively. (Although the M:N notation is a standard label for the many-to-many relationship, the label M:M may also be used.)

Mapping Cardinalities

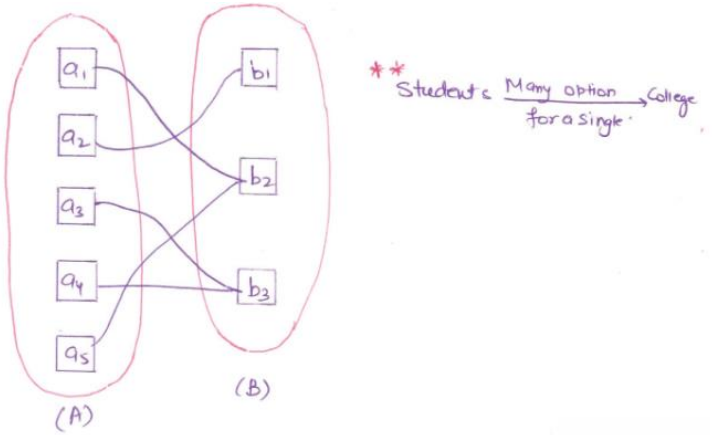
- The relationship set advisor, between the instructor and student entity sets may be One-to-one, one-to-many, many-to-one, or many-to-many.
- To distinguish among these types, we draw either a directed line (→) or an undirected line (—) between the relationship set and the entity set in question, as follows:
- **One-to-one Mapping Cardinality**
  - An entity in A is associated with "at most" One entity in B and an entity in B is associated with at most one entity in A.



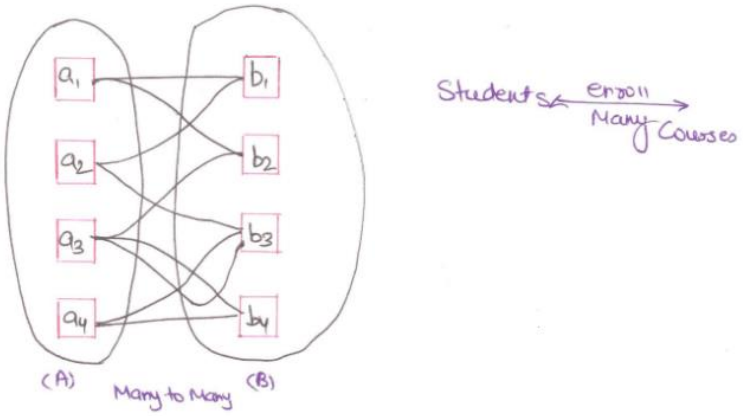
- **One to Many Mapping Cardinality**
  - An entity in A is associated with any number (Zero more) of entities B. An entity in B, however, can be associated with at most one entity in A.



- **Many to One Mapping Cardinality**
  - An entity in A is associated with at most one entity in B. An entity in B, however can be associated with any number (zero or more) of entities in A.



- **Many to Many Mapping Cardinality**
  - An entity in A is associated with any number(zero or more) of entities in B, and an entity in B is associated with any number (zero or more) of entities in A.



Constraints

- A constraint is a restriction placed on the data. Constraints are important because they help to ensure data integrity.
- Constraints are normally expressed in the form of rules.
- For example:
  - An employee's salary must have values that are between 6,000 and 350,000.
  - A student's GPA must be between 0.00 and 4.00.
  - Each class must have one and only one teacher.

Enhanced Entity Relationship Model (EER Model)

- EER is a high-level data model that incorporates the extensions to the original ER model.
- It is a diagrammatic technique for displaying the following concepts
  - Sub Class and Super Class
  - Specialization and Generalization
  - Union or Category
  - Aggregation
- These concepts are used when the comes in EER schema and the resulting schema diagrams called as EER Diagrams.

Features of EER Model

- EER creates a design more accurate to database schemas.
- It reflects the data properties and constraints more precisely.
- It includes all modeling concepts of the ER model.
- Diagrammatic technique helps for displaying the EER schema.
- It includes the concept of specialization and generalization.
- It is used to represent a collection of objects that is union of objects of different of different entity types.

SUB CLASS AND SUPER CLASS

- Sub class and Super class relationship leads the concept of Inheritance.
- The relationship between sub class and super class is denoted (d) with symbol.
- **Super Class**
  - Super class is an entity type that has a relationship with one or more subtypes.
  - An entity cannot exist in database merely by being member of any super class. For example: Shape super class is having sub groups as Square, Circle, Triangle.
- **Sub Class**
  - Sub class is a group of entities with unique attributes.
  - Sub class inherits properties and attributes from its super class. For example: Square, Circle, Triangle are the sub class of Shape super class

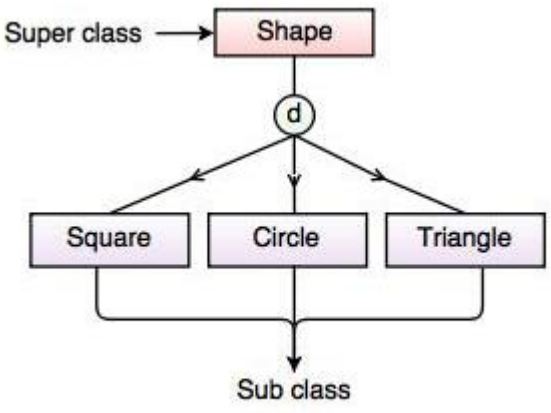
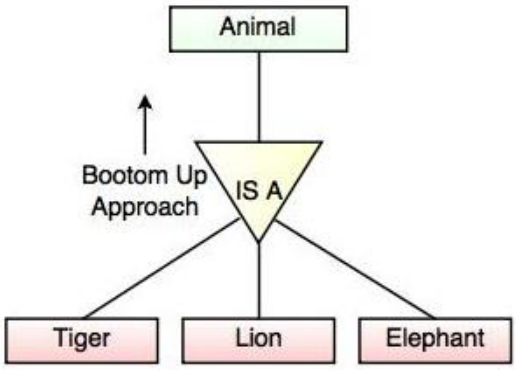


Fig. Super class/Sub class Relationship

SPECIALIZATION AND GENERALIZATION

- **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 approach, in which two lower level entities combine to form a higher level entity.
  - Generalization is the reverse process of Specialization.
  - It defines a general entity type from a set of specialized entity type.
  - It minimizes the difference between the entities by identifying the common features.
  - For Example:



- **Specialization**
  - Specialization is a process of categorizing or dividing a higher-level entity into multiple lower-level entities of a similar kind is known as specialization.
  - It is a top down approach, in which one higher entity can be broken down into two lower level entity.
  - It maximizes the difference between the members of an entity by identifying the unique characteristic or attributes of each member.
  - It defines one or more sub class for the super class and also forms the superclass/subclass relationship
  - For example:

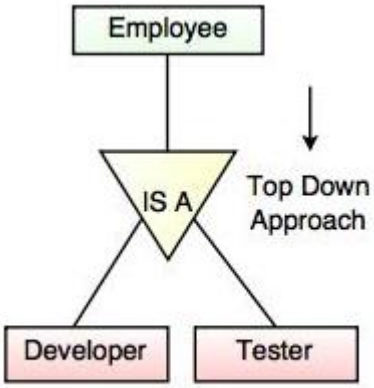


Fig. Specialization

AGGREGATION

- Aggregation in DBMS(Database Management System) is a process of combining two or more entities that cannot be used alone, to form a more meaningful entity.
- It abstracts a relationship between objects and viewing the relationship as an object.
- It is a process when two entity is treated as a single entity.

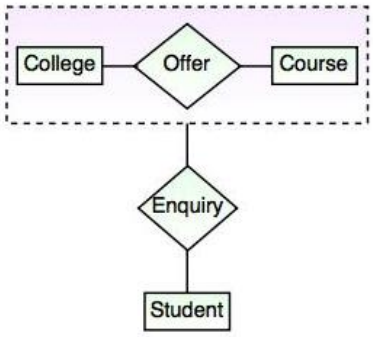


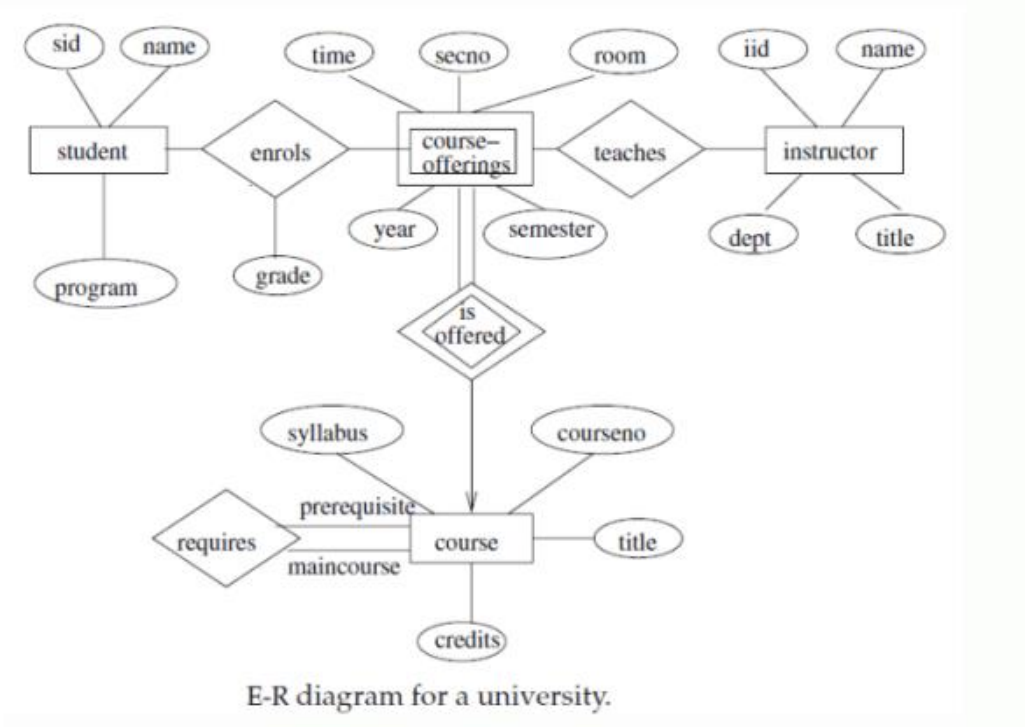
Fig. Aggregation

EXAMPLE OF ER DIAGRAM

Q.1) A university registrar's office maintains data about the following entities:

1. Courses, including number, title, credits, syllabus, and prerequisites;
2. Course offerings, including course number, year, semester, and section number, instructor(s), timings, and classroom;
3. Students, including student-id, name, and program;
4. Instructors, including identification number, name, department, and title.

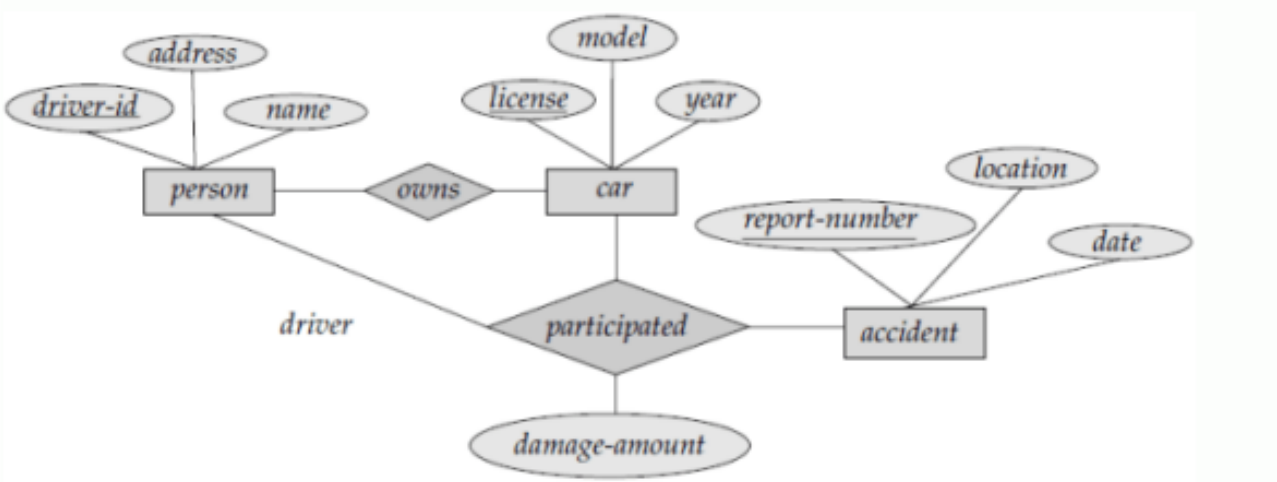
Further, the enrollment of students in courses and grades awarded to students in each course they are enrolled for must be appropriately modeled. Construct an E-R diagram



E-R diagram for a university.

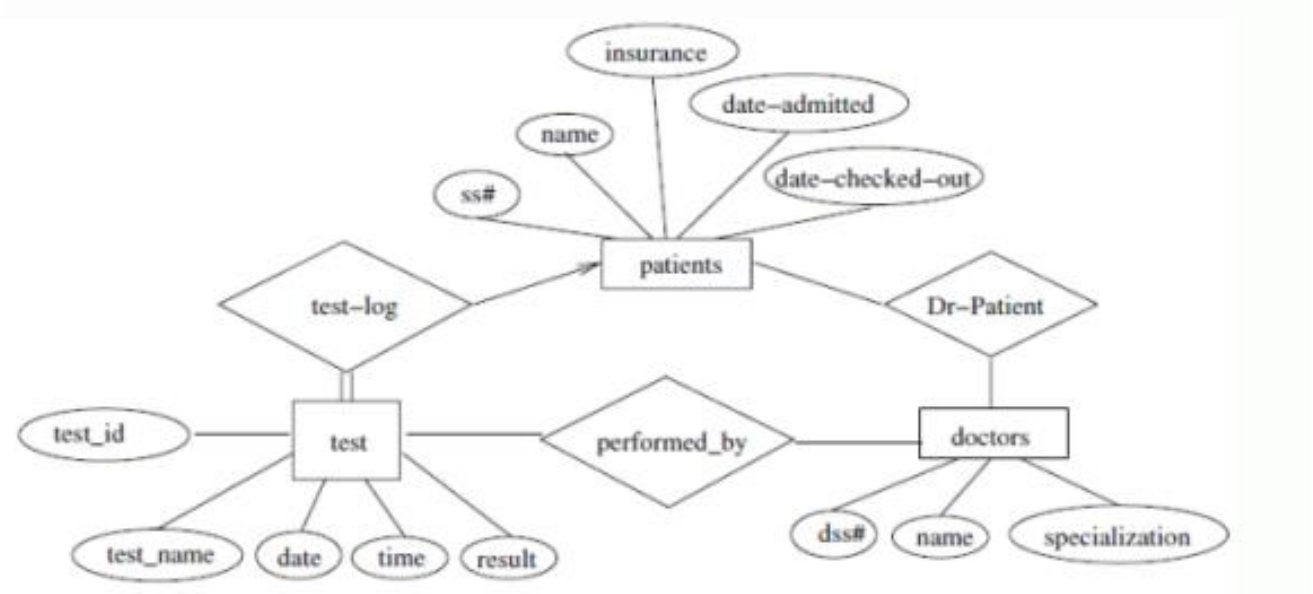
for the registrar's office. Document all assumptions that you make about the mapping constraints.

Q.2) 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.



E-R diagram for a Car-insurance company.

Q.3) Construct an E-R diagram for a hospital with a set of patients and a set of medical doctors. Associate with each patient a log of the various tests and examinations conducted.



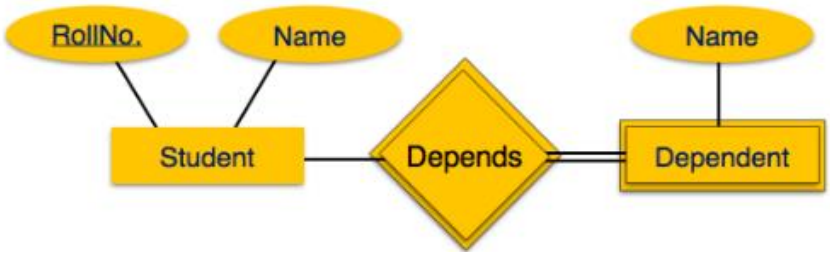
E-R diagram for a hospital.

Mapping from ER Model to Relational Model

- ER Model, when conceptualized into diagrams, gives a good overview of entity-relationship, which is easier to understand. ER diagrams can be mapped to relational schema, that is, it is possible to create relational schema using ER diagram. We cannot import all the ER constraints into relational model, but an approximate schema can be generated.
- There are several processes and algorithms available to convert ER Diagrams into Relational Schema. Some of them are automated and some of them are manual. We may focus here on the mapping diagram contents to relational basics.
- ER diagrams mainly comprise of –
  - Entity and its attributes
  - Relationship, which is association among entities.

Mapping Entity

- Create table for each entity.
- Entity's attributes should become fields of tables with their respective data types.
- Declare primary key.
- For each weak entity type with owner entity, create a table and include all simple attributes of weak entity type as column of table, including foreign key attributes as the primary key attribute of the table that correspond to he owner entity type.
  - Strong Entity



Student

RollNO(PRIMARY KEY)	NAME
1	Akash
2	Nehal
3	Omkar

  - Weak Entity

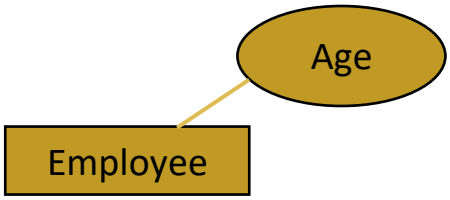
RollNO(PRIMARY KEY)	NAME	DNAME
1	Akash	Varun
2	Nehal	Chirag
3	Omkar	Shree

Attribute to column of table

1. Simple Attribute

- Simple attribute can be directly converted to a column in relational model.
- Example:

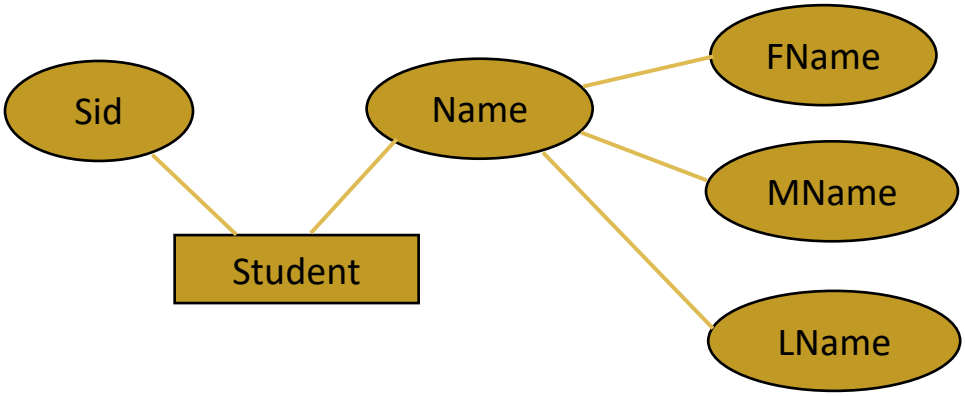
  Employee 'Age' can be directly converted to column.



EID	Age
1	23
2	24
3	25
4	29
5	27

2. Composite Attribute

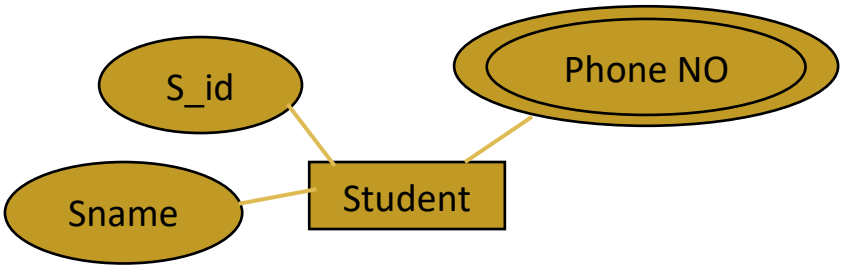
- These attributes need to be stored as set of simple component attributes in relational model by avoiding actual attribute.
- Example:



Student_id	S_Name	M_Name	L_Name
1	Harshad	Rupali	Malar
2	Bipin	Anand	Shinde
3	Anand	Ganesh	Panchal
4	Tushar	Bipin	Pimple

3. Multi Valued Attribute

Multi Values attributes are mapped as a relation which includes combination of the primary key of table and multi valued attribute as a composite primary key.



Student_id	S_Name
1	Harshad
2	Bipin
3	Anand
4	Tushar

Student_id	Mobile No
1	7895623
1	7842513
2	513564
2	7815633